Optimisations en Electrotechnique par Algorithmes Génétiques

Lucas Saludjian

THESE

présentée par

Lucas SALUDJIAN
(Ingénieur ENSEEIHT)

Pour obtenir le grade de **DOCTEUR**
de l'**INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE**
(Arrêté ministériel du 30 mars 1992)
(Spécialité : Génie Electrique)

* * *

Optimisations en Electrotechnique par Algorithmes Génétiques

* * *

Date de soutenance : **27 juin 1997**

Composition du jury :

Messieurs	A. NICOLAS	*Président et Rapporteur*
	B. TRANNOY	*Rapporteur*
	D. COLOMBANI	*Examinateur*
	P. TIXADOR	*Examinateur*
	J.L. COULOMB	*Examinateur*

Thèse préparée au sein du Laboratoire d'Électrotechnique de Grenoble

THESE

présentée par

Lucas SALUDJIAN
(Ingénieur ENSEEIHT)

Pour obtenir le grade de **DOCTEUR**

de l'**INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE**

(Arrêté ministériel du 30 mars 1992)

(Spécialité : Génie Electrique)

* * *

Optimisations en Electrotechnique par Algorithmes Génétiques

* * *

Date de soutenance : **27 juin 1997**

Composition du jury :

Messieurs	A. NICOLAS	*Président et Rapporteur*
	B. TRANNOY	*Rapporteur*
	D. COLOMBANI	*Examinateur*
	P. TIXADOR	*Examinateur*
	J.L. COULOMB	*Examinateur*

Thèse préparée au sein du Laboratoire d'Électrotechnique de Grenoble

REMERCIEMENTS

Remerciements

Je tiens tout d'abord à remercier Jean Louis COULOMB, professeur à l'Institut National Polytechnique de Grenoble, qui a assuré l'encadrement de mon travail. Il est souvent banal de remercier son directeur de thèse, mais il se trouve ici que je ne peux y échapper tant mes trois années passées à ses côtés m'ont procuré de très bon moments. Jean Louis, par sa gentillesse, sa simplicité et son authenticité a été une formidable source de motivation durant toute la thèse. Les discussions passionnantes et passionnées sur des thèmes n'ayant pas seulement trait au travail et qui se terminèrent souvent par de bonnes rigolades auront marqué mon passage au Laboratoire.

Je tiens aussi à remercier tout particulièrement Gérard MEUNIER, directeur de recherche au CNRS, pour l'atmosphère chaleureuse qu'il a su créer au sein de l'équipe modélisation. Sa décontraction, sa générosité (et ses talents de footballeur aussi ;-)) en font un personnage incontournable du laboratoire.

J'adresse également mes sincères remerciements :

Au Professeur Alain NICOLAS qui m'a fait l'honneur de présider le jury et qui a eu la tâche de juger cette thèse en tant que rapporteur.

Au Professeur Bernard TRANNOY qui m'a fait l'honneur d'être rapporteur de ce travail.

A Monsieur COLOMBANI, directeur technique de la société CEDRAT-RECHERCHE, qui a accepté de participer à mon jury

A Pascal TIXADOR, chargé de recherche au CNRS, qui a accepté d'être membre du jury. Je tiens aussi à le remercier sincèrement pour le temps qu'il a pu me consacrer à me faire découvrir le monde des basses températures et des supraconducteurs et pour la disponibilité et la gentillesse dont il a fait preuve pour répondre à mes questions de néophyte.

A Patrice LABIE, vieux montagnard de notre laboratoire que j'ai dû traîner bien malgré lui dans des lieux de perditions pour y boire des cafés et taper quelques discussions sur des domaines aussi divers et varié que Flux3d ou le dernier millésime des vins de la région ...

A Seddik BACHA avec qui j'ai eu le plaisir de discuter de long moments lors des pauses cigarettes et comme dit l'autre... El Thaoura Hata Naasr... hein Seddik !

A Nouredine HADJ-SAID pour le nombre d'heures qu'il a passé à m'expliquer les problèmes de réseaux mais surtout pour les discussions chaleureuses que nous avons pu avoir.

A Christian SCHAEFFER, Luc MEYSENC et Stéphane RAËL qui n'ont pas ménagé leur temps pour m'initier (moi pauvre débutant) aux méandres de l'électronique de puissance.

A Nicolas RETIERE ... sous ses aspects de rebelle associable se cache un homme d'une rare qualité humaine. Je ne compte plus les bringues que nous avons fait avec lui et Cécile et qui nous donnerons plus tard, lorsque nous serons complètement gaga, des histoires à raconter au coin du feu... Ah tu te souviens Grenoble ;-). Un seul regret tout de même ... après trois ans passés à me côtoyer sur les terrains, Nicolas ne sait toujours pas aligner 5 dribbles de suite au foot... quelle déception !

Comment ne pas citer Le ZG1 ! Toujours un métro de retard, bringueur hors catégories et d'une générosité sans égale ... il a été de toutes les fêtes et ce n'est pas fini hein Zg1 ?

Méfie toi des bretonnes ! Combien d'amis bretons me l'ont répété! Comme ils se trompent... Gwenaëlle LE_COAT est une délicieuse jeune fille (enfin pas trop jeune non plus...) dont la décontraction et la bonne humeur vous redonne le moral en un rien de temps !

J'ai eu aussi un vrai plaisir à côtoyer ces trois années Claire DIVOUX. Bonne humeur et franche rigolade ont souvent été au rendez-vous et je la remercie pour tous ces bons moments que nous avons pu avoir.

Tou as une clope ? Mais oui Christian GOLOVANOV ! Cette question a souvent été l'occasion de bons moments de détente et je remercie donc Christian de ne pas avoir acheté de cigarettes durant ces 3 années!

Enfin, je serais éternellement reconnaissant à la bibliothécaire de l'IMAG qui m'a toujours autorisé, et avec le sourire en plus !, à farfouiller dans les précieuses informations de sa bibliothèque.

Je ne peux malheureusement pas détailler la liste de toutes les personnes avec qui j'ai eu le plaisir de discuter mais je voudrais néanmoins citer :
Patrick GUILLOT, Marie-Thérèse LOUBINOUX, Florence FRANCOIS, Fabrice LEDORZE, Yves MARECHAL, Etiennette CALLEGHER, François-Xavier ZGAINSKI, Frédéric WURTZ, Stéphane BERGEON, DJIDJI, Jean Claude FIS, Catherine TALOWSKI, Till WELFONDER, Stéphane COURTINE, Severine GUFFON, Nadège PIETTE, Coralie COUTEL, Laurent GROS, Luiz DE MEDEIROS, Roland PACAUT, Joël CONRAD, Patrick EUSTACHE, Bruno FERRARI, Damien CHASSOULIER, Jerôme PECHOUX.

Je ne peux terminer cette liste sans remercier ma famille qui m'a soutenu tout au long de ces années ainsi que tous les amis avec qui j'ai eu le plaisir de partager de très bons moments à Grenoble.

TABLE DES MATIÈRES

Table des matières

Introduction Générale
p 5

PARTIE 1 : UN PEU DE THEORIE...

Chapitre I
Optimisation déterministe et stochastique
p 8

1 Introduction 8
2 Généralités 10
2.1 Formulation mathématique d'un problème d'optimisation 10
2.2 Optimum global et optima locaux 10
3 Méthodes déterministes 11
3.1 Rappels 11
3.1.1 Conditions nécessaires d'optimalité 11
3.1.2 Théorème de Kuhn et Tucker 1951 12
3.1.3 Conditions suffisantes de Kuhn et Tucker dans le cas convexe 12
3.2 Méthodes de Transformation 13
3.2.1 Méthodes de pénalités 13
A Méthode de pénalités extérieures 13
B Méthodes de pénalités intérieures 14
3.2.2 Méthode de Lagrangien Augmenté 15
A Problème à contraintes d'égalités 15
B Problème à contraintes d'inégalités 16
C Mise en oeuvre pratique de la méthode 17
3.3 Méthode d'optimisation sans contraintes 18
4 Méthodes stochastiques 21
4.1 Algorithmes Génétiques 21
4.1.1 Introduction 21
4.1.2 L'Algorithme génétique de Holland 23
A Représentation des paramètres 23
B Opérateur de sélection 24

C Opérateurs de croisement et mutation 25
D Théorie des schémas 27
4.1.3 Modifications sur les opérateurs 29
A Sélection 29
B Algorithmes d'échantillonnage 33
C Croisement 34
D Mutation 35
4.1.4 Adaptation et fonction objectif 37
4.2 Recuit Simulé 39
4.2.1 Introduction 39
4.2.2 Algorithme de Recuit Simulé 39
4.2.3 Plan de Recuit 41
A Température initiale 41
B Décroissance de la température 42
C Longueur des chaînes de Markov 42
D Création d'une nouvelle configuration dans la chaîne de Markov 43
E Critère d'arrêt 43
5 Validations 44
5.1 Fonctions testées et leurs principales caractéristiques 44
5.2 Tests effectués 47
5.3 Algorithme génétique 48
5.4 Recuit Simulé 52
5.5 Lagrangien Augmenté 53
6 Conclusion 54

Chapitre II
Vers un algorithme d'optimisation performant...
p 56

1 Introduction 56
2 Viabilité du codage binaire 58
3 Un Algorithme génétique avec codage réel 59
3.1 Présentation de l'algorithme mis au point 59
3.1.1 Codage des paramètres 59
3.1.2 Opérateurs de Croisement 59
3.1.3 Opérateurs de Mutation 60
3.2 Comparaison des performances avec celles du codage binaire 63
4 Hybridation 66
5 Parallélisation 70
6 Conclusion 73

PARTIE 2 : ... ET QUELQUES APPLICATIONS A L'ELECTROTECHNIQUE

Chapitre III

Optimisation d'un refroidisseur de puce IGBT

p 75

1 Introduction 75
2 Technologie des Micro-canaux 76
3 Etude Thermique du dispositif 77
4 Minimisation de la résistance thermique totale 80
5 Validation expérimentale 83
6 Conclusion 87

Chapitre IV

Optimisation de maillage CAO

p 88

1 Introduction et définition du problème 88
2 Généralités 91
2.1 Définitions 91
2.2 Maillage de Delaunay 92
3 Amélioration de maillage a priori 95
3.1 Introduction 95
3.2 Qualité d'un élément tétraédrique 96
3.3 Algorithmes d'amélioration de maillage 100
3.3.1 Outils de modification topologique 101
3.3.2 Algorithmes d'optimisation s'appuyant sur ces outils 104
3.3.3 Algorithmes d'amélioration non topologique 106
A Barycentrage 106
B Bougé de points 106
C Algorithme génétique pour l'optimisation de maillages 109
3.4 Tests effectués 114
4 Conclusion 125

Chapitre V
Optimisation de forme de structures électromagnétiques
p 126

1 Introduction et présentation du problème 126
2 Principe de la méthode éléments finis 128
2.1 Du phénomène physique étudié au modèle 128
2.2 De l'EDP au système matriciel- Méthode de Galerkin 130
2.3 Optimisation et C.A.O. 132
2.4 Pourquoi ne pas s'arrêter là ? 135
3 Calcul des dérivées d'ordre élevé et construction du développement de Taylor 136
3.1 Principe général 136
3.2 Quelques remarques 137
3.3 Dérivations de M et S par rapport aux paramètres physiques 138
3.3.1 Paramètre ν_k 138
3.3.2 Paramètre $\omega\sigma_k$ 138
3.3.3 Paramètre J_{sk} 139
3.4 Dérivation de M et S par rapport aux paramètres géométriques 139
4 Une nouvelle approche pour l'optimisation de forme 142
5 Problème Test 144
5.1 Généralités sur le SMES 144
5.2 Optimisation 146
5.3 Formulation mathématique du problème 148
6 Résultats 148
6.1 Le Logiciel Flux-param - Développements de Taylor obtenus 148
6.2 Validité du développement 151
6.3 Optimisation - Temps de calcul 152
7 Conclusion 155

Conclusion
p 156

Bibliographie
p 159

INTRODUCTION

Introduction

Dans tous les domaines des sciences, les ingénieurs sont amenés à concevoir de nouveaux dispositifs. Cette conception, au début manuelle et basée sur l'expérience et la pratique, devient de plus en plus automatisée grâce à l'arrivée des ordinateurs. En effet, l'introduction de "modèles numériques" autorise maintenant le concepteur à réaliser, par modifications successives, des dispositifs plus performants, tout en s'affranchissant du coût de construction des prototypes réels.

Cette démarche de l'ingénieur vers la conception d'un dispositif de grandes performances est grandement facilité et devient bien plus efficace dès lors que les modifications successives sont prises en charge par une méthode automatique d'optimisation plutôt que par des tâtonnements plus ou moins intuitifs. L'étude de telles méthodes automatiques, dans le domaine de l'électrotechnique, fait précisément l'objet de cette thèse.

Le chapitre I est surtout bibliographique et propose un état de l'art des méthodes d'optimisation. Certaines de ces méthodes sont dites *déterministes* car elles conduisent, pour une solution initiale donnée, toujours au même ré ultat final. Pour trouver l'optimum, elles s'appuient sur une direction de recherche qui est en général, fournie par les dérivées de la fonction à optimiser. Ces méthodes déterministes ont la réputation d'être efficaces mais elles ne peuvent malheureusement être employées que lorsque la solution initiale est proche de l'optimum recherché.
Ce point constitue un inconvénient majeur dans le cas d'une fonction à optimiser possédant *plusieurs optima,* et cela nous a poussé à étudier de près les méthodes d'optimisation dites *stochastiques.*
Ces méthodes, contrairement aux méthodes déterministes, avancent dans leur recherche en grande partie grâce à des mécanismes aléatoires. Elles constituent un bon moyen pour s'échapper des *optima locaux* et pour fournir *l'optimum global* du problème et ce, quelle que soit la solution initiale choisie.
Dans cette classe de méthodes, notre choix s'est arrêté sur les *algorithmes génétiques* dont les bases théoriques sont passionnantes et qui, simplement dit, miment le processus de la sélection naturelle observé par Darwin dans l'évolution des espèces.

Dans ce premier chapitre, nous nous attacherons à mettre en évidence les points forts et les points faibles des différentes méthodes d'optimisation. Pour cela, nous avons choisi de comparer leur comportement sur différentes fonctions tests de nature mathématique très variée et dont l'intérêt est de présenter, de façon isolée, les difficultés qui peuvent survenir dans les problèmes réels.

Dans le second chapitre, nous tenterons d'utiliser au mieux les constatations et les conclusions du premier chapitre pour construire un algorithme d'optimisation performant c'est-à-dire à la fois, capable de localiser l'optimum global et peu coûteux en nombre d'évaluations de la fonction à optimiser. Nous étudierons ainsi le couplage d'une méthode déterministe et d'un algorithme génétique. Nous verrons que le calcul parallèle offre aussi des perspectives très intéressantes dans le cas de l'algorithme génétique pour diminuer le coût des évaluations.
Mais dans ce chapitre, nous aborderons surtout un point qui nous a semblé fondamental pour les algorithmes génétiques et qui consiste à introduire des informations supplémentaires concernant la "nature" des paramètres du problème. En effet, dans l'algorithme génétique traditionnel tous les paramètres sont codés sous la forme d'une *chaîne binaire*. Ce codage, bien qu'universel, peut mener à une recherche peu efficace dans certains cas que nous évoquerons, et notamment pour l'optimisation de maillage sur laquelle nous avons travaillé au chapitre IV. Ces problèmes sont mentionnés car nous n'avons pas voulu limiter nos optimisations à un domaine bien particulier de l'électromagnétisme comme l'optimisation de forme où l'algorithme génétique traditionnel avait déjà été employé.

Les chapitres suivants sont consacrés à quelques applications en électrotechnique.

Le chapitre III présente l'*optimisation d'un radiateur de dissipation thermique pour les composants de puissance*. Cette étude semble a priori très éloignée du domaine de l'optimisation et pourtant elle a constituée une bonne "mise en chauffe" pour les méthodes d'optimisation sur un problème électrotechnique réel.

Le chapitre IV est consacré à l'*optimisation des maillages éléments finis* en trois dimensions.
Des logiciels comme FLUX3D, développé dans notre Laboratoire, constituent des outils très performants de conception assistée par ordinateur. Ils permettent de résoudre des équations aux dérivées partielles en ramenant le problème à la résolution d'un système d'équations algébriques grâce au découpage du domaine physique étudié. Ce découpage (le maillage), en éléments finis, a une influence très importante sur le bon conditionnement du problème et donc sur la qualité de la solution obtenue. Nous essaierons donc d'améliorer la qualité d'un maillage en utilisant entre autres une méthode originale basée sur les algorithmes génétiques et directement issue des idées du chapitre II.

Le chapitre V a pour objet l'*optimisation de forme des structures électromagnétiques* dans le domaine de la magnétostatique bidimensionnelle. Il s'agit ici, de trouver les bonnes valeurs des paramètres de conception d'un dispositif (géométriques et/ou physiques) afin de répondre à un certain

objectif (par exemple une répartition uniforme du champ) tout en satisfaisant des contraintes mécaniques ou physiques sur ces paramètres.

Le but est d'utiliser conjointement une méthode d'optimisation efficace avec une méthode de calcul de champs précise. Cependant, l'utilisation d'une méthode éléments finis pour le calcul des grandeurs électromagnétiques est très coûteuse en temps de calcul lorsque les paramètres sont modifiés un grand nombre de fois par l'algorithme d'optimisation. Nous nous sommes donc tournés vers une méthode d'évaluation basée sur le *développement de Taylor* d'une solution éléments finis obtenue pour des valeurs initiales des paramètres. Ce développement constitue en fait, une approximation de la solution qui demande un investissement initial important (calculs de dérivées d'ordre élevé) mais qui autorise ensuite des évaluations, en théorie, instantanées. Le couplage de cette méthode d'évaluation avec un algorithme génétique constitue une nouvelle approche de l'optimisation de forme et fait l'objet de l'étude du chapitre V.

PARTIE 1

Un peu de Théorie...

Chapitre I

Optimisation déterministe et stochastique

Chapitre I

Optimisation déterministe et stochastique

1 INTRODUCTION

Les problèmes d'optimisation remontent à des temps aussi anciens que les mathématiques. Pour mémoire, les premières techniques de résolution sont attachées à des noms comme Newton, Lagrange ou Cauchy. Cependant, durant ces 30 ou 40 dernières années, d'importants progrès ont été réalisés grâce notamment à l'arrivée des ordinateurs. Ceux-ci ont permis de résoudre dans des temps très courts, des problèmes de plus en plus complexes, dans de très nombreux domaines de la science. Dans le domaine automobile, on cherche à optimiser la forme des voitures pour les rendre plus aérodynamiques. Dans le domaine des télécommunications ou de l'informatique, c'est la configuration du réseau que l'on cherche à optimiser. Ce ne sont là que quelques exemples bien connus. En électromagnétisme, il s'agira par exemple de trouver une configuration des paramètres de conception d'une machine afin d'obtenir une distribution de champ donnée [Simkin92] [DeVasconcelos94a] [Kadded92].

Les méthodes d'optimisation sont très nombreuses. On peut toutefois les classer en deux grandes familles : les méthodes *déterministes* et les méthodes *stochastiques*.

Les premières, comme leur nom l'indique, ne laissent aucune place au hasard et conduiront pour un contexte initial donné à une même solution finale. Pour ces méthodes, l'exploration de l'espace des solutions se fait grâce à la connaissance d'une direction de recherche qui peut être donnée par le gradient. Ce sont en général des méthodes efficaces, peu coûteuses, mais qui nécessitent en contrepartie une première solution connue proche de la solution optimale.

A l'opposé des méthodes déterministes, les méthodes stochastiques explorent l'espace des solutions grâce en partie à des mécanismes de transitions aléatoires. Ainsi, plusieurs exécutions successives de ces méthodes, pourront conduire à des résultats différents (pour une même solution initiale!). Cependant ces méthodes se révèlent très intéressantes pour plusieurs raisons. Elles ne requièrent comme seule information du problème que la valeur de la fonction à optimiser et ne nécessitent donc pas la connaissance du gradient. Cela constitue un avantage dans le cas de problèmes à paramètres discrets ou lorsque la valeur de la fonction nécessite une résolution numérique. L'autre grand intérêt est la capacité qu'ont une majorité de ces algorithmes stochastiques à trouver l'optimum global. En contrepartie, ces méthodes demandent un nombre important d'évaluations avant d'atteindre l'optimum

global, ce qui, dans le cas d'une fonction évaluée par un logiciel éléments finis par exemple, peut entraîner un coût en temps de calcul rapidement prohibitif.

Ce problème peut être en partie levé par une programmation parallèle, tout particulièrement dans le cas de l'algorithme génétique. L'hybridation d'un algorithme stochastique avec une méthode déterministe peut constituer un autre remède : l'algorithme stochastique se chargeant alors de localiser un point dans la région de l'optimum global et l'algorithme déterministe s'efforçant de converger rapidement et à moindre coût vers cet optimum.

Dans ce chapitre nous étudierons les deux types de méthodes. Parmi les méthodes déterministes, c'est la méthode du Lagrangien Augmenté qui a retenu notre attention. Elle est aujourd'hui une des méthodes les plus robustes pour résoudre les problèmes non linéaires [Minoux83]. Elle a été employée récemment en mécanique [Sunar91] ainsi qu'en électromagnétisme pour l'optimisation de forme [DeVasconcelos94b]. Cette méthode se base en fait sur la transformation d'un problème sous contraintes en une suite de problèmes sans contraintes. Pour traiter ces derniers nous emploierons une méthode du type Quasi-Newton qui nécessite la connaissance du gradient et d'une approximation de la dérivée d'ordre 2.

En ce qui concerne les méthodes stochastiques, notre choix s'est arrêté sur le Recuit Simulé et surtout sur les Algorithmes Génétiques. Ces derniers ont fait l'objet de nombreux développements ces dernières années. Mais jusqu'à présent, en électromagnétisme, ils n'ont été utilisés que dans leur forme la plus simple. Nous verrons donc comment prendre en compte ses dernières évolutions.

Enfin, nous terminerons par une étude comparative des divers algorithmes mis au point. Le but étant de pouvoir tirer profit des forces de chacun d'entre eux pour pouvoir construire un algorithme d'optimisation performant.

2 GENERALITES

2.1 Formulation mathématique d'un problème d'optimisation

Un problème d'optimisation peut être écrit sous la forme :

$$(P)\begin{cases} Min\, f(\boldsymbol{x}) \in \Re \\ g_i(\boldsymbol{x}) \le 0 & i=1,\ldots,m \\ h_j(\boldsymbol{x}) = 0 & j=1,\ldots,l \\ x_k^{min} \le x_k \le x_k^{max} & k=1,\ldots,n \end{cases} \qquad (I.1)$$

$\boldsymbol{x}$ est un vecteur de n composantes x_k qui sont les inconnues de notre problème. En électrotechnique x_k peut aussi bien représenter des grandeurs physiques (courant, induction, ...) que des paramètres de conception (dimensions géométriques, nombre de spires de bobinage, ...).
f est le critère à minimiser appelé aussi objectif et les fonctions (g_i) et (h_j) représentent les contraintes d'inégalités et d'égalités. Les contraintes x_k^{min} et x_k^{max} qui bornent les variations des inconnues x_k s'appellent des contraintes de domaine et sont en général formulées sous forme de contraintes d'inégalités : $x_k^{min} - x_k \le 0$ et $x_k - x_k^{max} \le 0$.
Dans la pratique, on classifie les différents problèmes d'optimisation suivant la nature mathématique de la fonction objectif et de l'espace des contraintes. Les fonctions f, (g_i), (h_j) peuvent être continues ou discontinues, linéaires ou non linéaires, convexes ou non convexes, différentiables ou non différentiables ... Dans la suite -sauf mention contraire- on ne considérera, pour simplifier les écritures, que les contraintes d'inégalités. En effet la contrainte d'égalité $h_j(\boldsymbol{x})=0$ est équivalente à: $h_j(\boldsymbol{x}) \le 0$ et $-h_j(\boldsymbol{x}) \le 0$.
Enfin, on notera $C = \{\boldsymbol{x} \in \Re^n / g_i(\boldsymbol{x}) \le 0 \quad i=1,\ldots,m\}$ l'espace des contraintes.

2.2 Optimum global et optima locaux

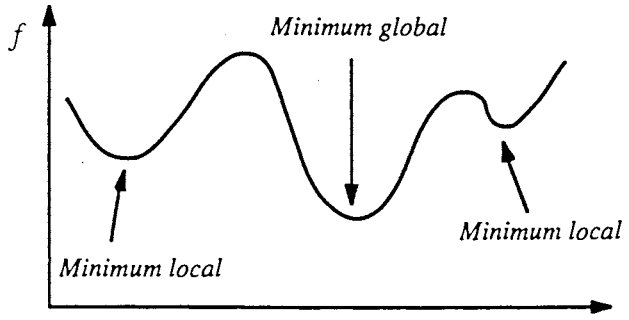


Figure I.1: Optimum local et global

Définition : • Un point $\boldsymbol{x}^* \in C$ est un **minimum local** s'il existe un voisinage de $\boldsymbol{x}^*$ noté $V(\boldsymbol{x}^*)$ tel que $\forall\, \boldsymbol{x} \in V(\boldsymbol{x}^*)\ f(\boldsymbol{x}^*) \le f(\boldsymbol{x})$.

- Un point $x^* \in C$ est un **minimum global** si $\forall x \in C \; f(x^*) \leq f(x)$.

Comme nous le verrons, dans le cas des méthodes d'optimisation déterministes, il est souvent aisé de caractériser les optima locaux par des conditions nécessaires et suffisantes. Par contre, il est en général impossible de caractériser l'optimum global d'un problème d'optimisation sauf dans le cas très particulier des problèmes convexes. Malheureusement, en Electrotechnique, les problèmes rencontrés ne présente pas en général cette propriété de convexité si appréciable [Gottvald92] [DeVasconcelos94b] [Magele93] [Magele96] [Üler94]. Les méthodes déterministes que nous allons présenter, ne nous assurerons donc a priori que de l'existence d'optima locaux.

3 METHODES DETERMINISTES

Les méthodes déterministes peuvent être séparées en deux grandes classes. La première regroupe toutes les méthodes dites d'ordre 0 c'est-à-dire qui ne nécessitent pas la connaissance de la dérivée première (et a fortiori des dérivées d'ordre supérieur à 1). Ces méthodes sont en général peu précises et convergent très lentement vers l'optimum local [Kowalik68]. Mais elles offrent l'avantage de se passer du calcul des gradients ce qui peut être intéressant lorsque la fonction n'est pas différentiable ou lorsque leurs calculs nécessitent un coût important. D'une manière générale, de telles méthodes sont employées en début d'exécution pour repérer la région de l'optimum local. L'autre classe, par laquelle nous commencerons la présentation, suppose que toutes les fonctions f, g_i, h_j sont continûment différentiables. Cette propriété permet d'exploiter au mieux une information très importante sur la direction de recherche : le **gradient**.

Cette section débutera par des rappels théoriques, puis nous présenterons diverses méthodes de résolution dans le cadre des problèmes sous contraintes en accordant une place toute particulière à la méthode du Lagrangien Augmenté.

3.1 Rappels

3.1.1 Conditions nécessaires d'optimalité

On s'intéresse ici au problème (P) :

$$(P) \begin{cases} Min\, f(x) \\ g_i(x) \leq 0 \quad i = 1,\ldots,m \\ x \in \Re^n \end{cases} \tag{I.2}$$

Définition

On appelle fonction de Lagrange associée au problème (P) la fonction :

$$L(x,\lambda) = f(x) + \sum_{i=1}^{i=m} \lambda_i g_i(x) \tag{I.3}$$

où $\lambda_i \geq 0$ sont appelés multiplicateurs de Lagrange.

3.1.2 Théorème de Kuhn et Tucker 1951

Une condition nécessaire pour que x^* soit un optimum local de (P) est qu'il existe $\lambda^* = (\lambda_i^*)_{i=1,m}$ appelés multiplicateurs de Kuhn et Tucker tel que :

$$\begin{cases} \nabla f(x^*) + \sum_{i=1}^{i=m} \lambda_i^* \nabla g_i(x^*) = 0 \\ \lambda_i^* g_i(x^*) = 0 \quad \forall i = 1,\ldots,m \end{cases} \tag{I.4}$$

qui s'écrit aussi à l'aide de (I.3) :

$$\begin{cases} \nabla_x L(x^*, \lambda^*) = 0 \\ \lambda_i^* g_i(x^*) = 0 \quad \forall i = 1,\ldots,m \end{cases} \tag{I.5}$$

Dans le cas où figurent uniquement des contraintes d'égalités on obtient les conditions de Lagrange : il existe des multiplicateurs $(\mu_j^*)_{j=1,l}$ de *signes quelconques* tels que :

$$\nabla f(x^*) + \sum_{j=1}^{j=l} \mu_j^* \nabla h_j(x^*) = 0 \tag{I.6}$$

3.1.3 Conditions suffisantes de Kuhn et Tucker dans le cas convexe

Si les fonctions f et (g_i) sont convexes alors x^* est un optimum global de (P) *si et seulement si* les conditions de Kuhn et Tucker en x^* sont vérifiées.
L'hypothèse de convexité est en fait assez restrictive. Par exemple, pour des problèmes d'optimisation de forme en électrotechnique, cette hypothèse n'est pas vérifiée sur tout le domaine, mais uniquement sur un domaine réduit [Kadded93]. On pourra alors se contenter d'une hypothèse de convexité locale pour déduire une condition suffisante d'optimalité locale. Ainsi x^* est un optimum local de (P) *si et seulement si* :

- les fonctions f et (g_i) sont convexes dans un voisinage de x^*.
- les conditions de Kuhn et Tucker sont vérifiées en x^*.

Les principales méthodes de gradients sont constituées de deux grandes classes. La première de ces classes regroupe les méthodes dites primales. Ces méthodes opèrent directement sur le problème à résoudre, sans le modifier. Elles possèdent l'avantage de fournir à chaque itération une solution approchée réalisable c'est-à-dire qui satisfait les contraintes [Minoux83]. Nous ne les aborderons pas car ces méthodes se révèlent être moins efficaces et moins robustes que la seconde classe de méthodes: celles qui transforment le problème initial pour le ramener à la résolution d'une suite de problèmes sans contraintes.

3.2 Méthodes de Transformation

3.2.1 Méthodes de pénalités

L'intérêt de ces méthodes est la simplicité de leurs principes et leur relative efficacité pratique [Ryan74] [Minoux83]. Le concept de base est de transformer la résolution du problème (P) sous contraintes en une suite de résolutions de problèmes sans contraintes en associant à l'objectif une pénalisation dans l'évaluation dès qu'une des contraintes est violée. Schématiquement :

Soit (P) le problème d'optimisation $\begin{cases} Min\, f(\boldsymbol{x}) \\ g_i(\boldsymbol{x}) \le 0 \quad i=1,\ldots,m \\ \boldsymbol{x} \in \Re^n \end{cases}$

Soit la fonction $P : \Re^n \rightarrow \Re$

$$P(y)=0 \text{ si } y \le 0 \qquad (I.7)$$
$$P(y)=+\infty \text{ si } y>0$$

et soit le problème sans contraintes (P1) :

$$(P1) \quad \underset{x \in \Re^n}{Min}\, \Phi(\boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{i=m} P(g_i(\boldsymbol{x})) \qquad (I.8)$$

alors la résolution du problème (P1) est équivalente à la résolution du problème initial (P). La fonction $\sum_{i=1}^{i=m} P(g_i(\boldsymbol{x}))$ se nomme fonction de pénalisation. Malheureusement, en pratique, cette transformation n'est pas applicable directement. En effet, la discontinuité de la fonction de pénalisation empêche l'utilisation des méthodes d'optimisation sans contraintes classiques que nous verrons plus loin. Mais ceci constitue les principes de base des méthodes de pénalités et en particulier de la méthode de pénalité extérieure [Fiacco68] et de la méthode de pénalité intérieure [Caroll61] que nous allons exposer.

A Méthode de pénalités extérieures

Soit (P) le problème $\begin{cases} Min\, f(\boldsymbol{x}) \\ g_i(\boldsymbol{x}) \le 0 \quad i=1,\ldots,m. \\ \boldsymbol{x} \in \Re^n \end{cases}$

Le problème (P) peut être remplacé par le problème sans contraintes [Fiacco68]:

$$(P1) \quad \underset{x}{Min}\, \Phi(\boldsymbol{x},r) = f(\boldsymbol{x}) + r \sum_{i=1}^{i=m} \left[max(0, g_i(\boldsymbol{x})) \right]^2 \qquad (I.9)$$

où $r>0$ est appelé coefficient de pénalité.

Le problème (P1) pourrait être résolu directement pour une valeur de r suffisamment grande de telle façon que les contraintes soient satisfaites mais ce choix entraîne un mauvais conditionnement de Φ et

donc engendre des problèmes numériques lors de la résolution [Minoux83]. C'est pourquoi les méthodes de pénalités sont en général résolues de manière itérative : une suite de valeurs croissantes de r est générée et à chaque étape k du processus on résout le problème d'optimisation sans contraintes :

$$(P1^k) \quad \underset{x}{Min}\,\Phi(x,r^k) = f(x) + r^k \sum_{i=1}^{i=m} \left[max(0, g_i(x))\right]^2 \qquad (I.10)$$

Lorsque k tend vers l'infini ($P_1{}^k$) devient équivalent à notre problème initial (P).

L'avantage principal de cette méthode est que le point initial peut être situé en dehors du domaine des contraintes $C = \{x \in \Re^n / g_i(x) \leq 0 \quad i = 1,\ldots,m\}$. Par contre l'inconvénient majeur est que la suite de solutions approchées x^k n'appartient pas à C : la solution est dite *approchée par l'extérieur de C*. Par conséquent, si pour quelque raison que ce soit, le processus est stoppé avant convergence nous nous retrouverions avec une solution approchée x^k qui serait non réalisable. Pour cette raison des méthodes dites de pénalités intérieures, pour lesquelles l'optimum est approché par l'intérieur de C, ont été mises en œuvre.

B Méthodes de pénalités intérieures

Le problème (P) est remplacé [Caroll61] par :

$$(P2) \quad \underset{x}{Min}\,\Phi(x) = f(x) - \frac{1}{r}\sum_{i=1}^{i=m} \frac{1}{g_i(x)} \qquad (I.11)$$

où $r > 0$ est le facteur de pénalité et $B(x) = \sum_{i=1}^{i=m} \frac{-1}{g_i(x)}$ est appelé fonction barrière.

Lorsque x appartient à C on a $B(x) > 0$ et lorsque x tend vers la frontière de C, $B(x) \rightarrow +\infty$. On ne peut donc jamais franchir la frontière de C au cours d'un processus de minimisation. Par conséquent, pour un point initial donné dans C, le processus itératif va générer une suite de solutions approchées x^k qui seront toutes réalisables. Néanmoins, la recherche d'un point initial dans C n'est pas aisée et il est souvent nécessaire d'avoir recours à un algorithme supplémentaire pour le trouver.

Les méthodes que nous venons de décrire présentent certains inconvénients. Notamment lorsque $r^k \rightarrow \infty$ la fonction Φ peut être mal conditionnée ce qui entraîne une convergence lente [Luenberger71]. Mais leur simplicité pratique nous permettra de les utiliser en conjonction avec les algorithmes génétiques pour traiter les problèmes sous contraintes. Nous allons maintenant présenter une méthode qui permet d'éliminer en partie les difficultés des méthodes de pénalités en assurant notamment une convergence vers l'optimum sans que le paramètre de pénalité tende vers l'infini. Ces méthodes sont appelées méthode de Lagrangien Augmenté [Powell69] et sont réputées pour être robustes et efficaces [Belegundu85].

3.2.2 Méthode de Lagrangien Augmenté

Nous présenterons ici la méthode dans le cas des contraintes d'égalité -Approche Hestenes et Powell [Powell69][Hestenes69]- puis nous généraliserons au cas des contraintes d'inégalités -Approche de Rockaffelar [Rockaffelar73]- Nous ne détaillerons pas ici toutes les formules de réactualisation des paramètres de l'algorithme et nous nous bornerons aux principes de la méthode. Le lecteur intéressé pourra trouver une description complète des diverses variantes dans l'ouvrage de [Minoux83] ou dans la thèse de [DeVasconcelos94b].

A Problème à contraintes d'égalités

$$\text{Soit (P)} \begin{cases} Min\, f(\boldsymbol{x}) \\ h_j(\boldsymbol{x}) = 0 \quad j = 1,\ldots,l \\ \boldsymbol{x} \in \Re^n \end{cases} \tag{I.12}$$

On rappelle que si $\boldsymbol{x}^*$ est un minimum de f, il existe des multiplicateurs de Lagrange de signes quelconques tel que la condition de Lagrange soit satisfaite c'est-à-dire:

$$\exists \left(\mu_j^*\right)_{j=1,l} \in \Re^l \text{ tels que } \nabla f(\boldsymbol{x}^*) + \sum_{j=1}^{j=l} \mu_j^* \nabla h_j(\boldsymbol{x}^*) = 0$$

$$\text{On pose } L(\boldsymbol{x},\lambda,r) = f(\boldsymbol{x}) + \sum_{j=1}^{j=l} \lambda_j h_j(\boldsymbol{x}) + r \sum_{j=1}^{j=l} h_j^2(\boldsymbol{x}) \tag{I.13}$$

avec $r > 0$ et $\lambda = (\lambda_1,\ldots,\lambda_l) \in \Re^l$.

La fonction L s'appelle la **fonction Lagrangienne augmentée**. On distingue clairement dans cette somme la fonction lagrangienne classique augmentée par une fonction de pénalité extérieure.

Principe de la méthode

Le principe de la méthode consiste à résoudre, de façon itérative, le problème sans contraintes augmenté $Min\, L(\boldsymbol{x},\lambda,r)$. On génère ainsi une suite de points $\boldsymbol{x}^k$ qui réalisent à chaque itération k le problème:

$$\underset{\boldsymbol{x}\in\Re^n}{Min}\, L(\boldsymbol{x},\lambda^k,r) = f(\boldsymbol{x}) + \sum_{j=1}^{j=l} \lambda_j^k h_j(\boldsymbol{x}) + r \sum_{j=1}^{j=l} h_j^2(\boldsymbol{x})$$

Une condition nécessaire pour que $\boldsymbol{x}^k$ soit un minimum local est que $\nabla_x L(\boldsymbol{x},\lambda^k,r) = 0$ (*Condition nécessaire d'optimalité d'un problème sans contraintes*) .

$$\text{soit que : } \nabla f(\boldsymbol{x}^k) + \sum_{j=1}^{j=l} \lambda_j^k \nabla h_j(\boldsymbol{x}^k) + 2r \sum_{j=1}^{j=l} \left[\nabla h_j(\boldsymbol{x}^k)\right]^T \nabla h_j(\boldsymbol{x}^k) = 0 \tag{I.14}$$

En utilisant la formule de réactualisation des multiplicateurs de Lagrange proposée par Hestenes:

$$\lambda_j^{k+1} = \lambda_j^k + 2rh_j(x^k) \text{ pour } j = 1,\ldots,l \tag{I.15}$$

il vient que $\nabla f(x^k) + \sum_{j=1}^{j=l} \lambda_j^{k+1} \nabla h_j(x^k) = 0$ (I.16)

A l'optimum (x^*, λ^*) on a donc $\nabla f(x^*) + \sum_{j=1}^{j=l} \lambda_j^* \nabla h_j(x^*) = 0$. De plus, à l'optimum, (I.15) s'écrit $\lambda_j^* = \lambda_j^* + 2rh_j(x^*)$. On en déduit donc que $h_j(x^*) = 0 \ \forall j$. En conclusion, si (x^*, λ^*) est la solution du problème sans contraintes de Lagrangien augmenté alors il vérifie :

$$\begin{cases} \nabla_x L(x^*, \lambda^*) = 0 \\ h_j(x^*) = 0 \ \ j = 1,\ldots,l \end{cases}$$

c'est-à-dire (x^*, λ^*) vérifie les conditions nécessaires d'optimalité de Lagrange du problème (P).

Intérêt de la méthode

Le conditionnement numérique du problème sans contraintes est en fait grandement amélioré par l'utilisation du Lagrangien augmenté . En effet, quelles que soient les valeurs prises par r, le gradient $\nabla_x L(x, \lambda^*, r)$ est nul en x^* puisque $\forall j \ \ h_j(x^*) = 0$. Par contre dans le cas d'une méthode de pénalités extérieures où on minimise $\Phi(x, r) = f(x) + r\sum_{j=1}^{j=l} h_j^2(x)$, le gradient $\nabla_x \Phi(x, r)$ en x^* est égal à $\nabla f(x^*)$ donc non nul en général quelle que soit la valeur de r. C'est pour cette raison que dans les méthodes de pénalités, r doit être pris le plus grand possible afin d'approcher le plus possible x^*, avec l'inconvénient que la fonction Φ devienne mal conditionnée.

B Problème à contraintes d'inégalités

Soit (P) $\begin{cases} Min\, f(x) \\ g_i(x) \le 0 \ \ i = 1,\ldots,m \\ x \in \Re^n \end{cases}$

L'extension des idées précédentes aux contraintes d'inégalités a été suggérée par [Rockaffelar73].
Le problème (P3) est équivalent au problème (I.12) sous contraintes d'égalités par ajout de variables d'écart positives.

$$\text{(P3)} \quad \begin{cases} Min\, f(x) \\ g_i(x) + s_i = 0 \ \ i = 1,\ldots,m \\ s_i \ge 0 \ \ et \ \ x \in \Re^n \end{cases} \tag{I.17}$$

La fonction Lagrangienne augmentée s'écrit donc :

$$L(x,\lambda,r)=f(x)+\sum_{i=1}^{i=m}\lambda_i(g_i(x)+s_i)+r\sum_{i=1}^{i=m}(g_i(x)+s_i)^2 \qquad (I.18)$$

où $r>0$.

$L(x,\lambda,r)$ doit être minimisée par rapport à x et au vecteur $s=(s_i)_{i=1,n}\geq 0$ des variables d'écart. La minimisation par rapport à s, pour x fixé, peut être effectuée analytiquement pour chaque s_i. En effet on a :

$$\frac{\partial L}{\partial s_i}=rs_i^2+(2rg_i(x)-\lambda_i)s_i=0 \text{ (Condition nécessaire d'optimalité)}$$

donc si $r>0$ le minimum est atteint pour $\begin{cases} s_i^*=-g_i(x)-\dfrac{\lambda_i}{2r} & si\ \ g_i(x)\leq\dfrac{\lambda_i}{2r} \\ s_i^*=0 & sinon \end{cases}$

On se ramène donc à un problème ne dépendant que des variables x en posant [Rockaffelar73] :

$$L(x,\lambda,r)=f(x)+\sum_{i=1}^{i=m}G(g_i(x),\lambda_i,r)$$

avec $G(g_i(x),\lambda_i,r)=\begin{cases} -\dfrac{\lambda_i^2}{4r} & si\ g_i(x)\leq-\dfrac{\lambda_i}{2r} \\ \lambda_i g_i(x)+r[g_i(x)]^2 & si\ g_i(x)\geq-\dfrac{\lambda_i}{2r} \end{cases}$

La réactualisation des multiplicateurs de Lagrange utilisée pourra être une variante de celle utilisée par Hestenes :

$$\lambda_i^{k+1}=Max\left(0,\lambda_i^k+2rg_i(x^k)\right)$$

C Mise en oeuvre pratique de la méthode

Dans la pratique, pour programmer une méthode de Lagrangien augmenté, on procède à chaque itération k, de la manière suivante :

(1) Pour λ^k et r^k donnés, trouver x^{k+1} minimum du problème sans contraintes $Min\,L(x,\lambda^k,r^k)$.
(2) Réactualiser λ^k
(3) Réactualiser r^k

Formules de réactualisation de λ^k

Il existe de nombreuses autres formules de réactualisation [Fletcher74] dont certaines basées sur les dérivées premières ou secondes.

Par exemple [Buys72],

$$\lambda^{k+1} = \lambda^k + \left[[\nabla g]^T \left[\nabla_x^2 L\right]^{-1} [\nabla g]\right]^{-1} g \text{ où } g = [g_1, \ldots, g_m]^T \quad \text{(I.19)}$$

L'inconvénient majeur d'une telle formule réside justement dans le calcul les dérivées premières et secondes ce qui ne la rend pas très simple à utiliser en ingénierie [Arora91][DeVasconcelos94a]. Cependant, lorsque la minimisation du problème sans contraintes $Min\, L(\boldsymbol{x}, \lambda^k, r^k)$ est résolu par des méthodes Quasi-Newton, il est intéressant d'utiliser toute l'information obtenue et notamment l'approximation H de l'inverse du Hessien $\left[\nabla_x^2 L\right]^{-1}$, ce qui permettra la mise en oeuvre de la procédure de réactualisation :

$$\lambda^{k+1} = \lambda^k + \left[[\nabla g] H [\nabla g]^T\right]^{-1} g \quad \text{(I.20)}$$

Paramètres de pénalité

Après la résolution à chaque itération du problème en $\boldsymbol{x}$ et la réactualisation des multiplicateurs, il faut mettre à jour le coefficient r. Celui-ci joue pratiquement le même rôle que dans les méthodes de pénalités. Selon les méthodes, la mise à jour peut prendre la forme:

$$r^{k+1} = \rho r^k \text{ avec } \rho > 1$$

Comme précisé plus haut, il est inutile d'utiliser des coefficients de pénalité trop élevées. Ainsi on évite le mauvais conditionnement numérique du problème sans contraintes $\underset{x}{Min}\, L(\boldsymbol{x}, \lambda^k, r^k)$.

Cependant, la formule de réactualisation des multiplicateurs de Lagrange converge vers λ^* avec un taux de convergence d'autant meilleur que r est grand [Luenberger71]. Il y a donc un compromis à trouver entre conditionnement du problème sans contraintes et la convergence des multiplicateurs de Lagrange. Ceci résulte dans le choix de valeurs *initiales* de r faibles et dans l'utilisation d'une méthode de gradient conjuguée ou Quasi-Newton pour la minimisation du problème sans contraintes (les méthodes de plus fortes pentes sont à proscrire [Minoux83])

3.3 Méthode d'optimisation sans contraintes

Les méthodes de transformations présentées demandent à chaque étape la résolution d'un problème sans contrainte. Pour les raisons évoquées précédemment - conditionnement du problème -, la minimisation du problème sans contrainte doit se faire par une méthode de gradient conjugué ou une méthode Quasi-Newton. Cette dernière possède l'avantage de nécessiter approximativement n fois moins d'étapes que les méthodes de gradient conjugué pour un même comportement asymptotique - taux de convergence - [McCormik72][Dennis74]. De plus, l'information obtenue lors de cette minimisation sans contraintes permettra la mise en oeuvre d'une formule de réactualisation des multiplicateurs de Lagrange du type (I.20).

Méthode Quasi-Newton

Cette méthode suppose que la fonction f - notre Lagrangien augmenté - est 2 fois continûment différentiable. Elle peut être vue comme une extension de la méthode de Newton c'est-à-dire une méthode dans laquelle la direction de recherche - de descente - s'exprime à l'aide d'une formule:

$$x^{k+1} = x^k - \alpha H^k \nabla f(x^k) \tag{I.21}$$

où H^k représente une approximation de l'inverse du Hessien de f (dans la méthode de Newton $H^k = \nabla^2 f(x^k)$) et $\alpha > 0$ le pas de descente.

Broyden, Fletcher, Goldfard, Shanno [Broyden70] [Fletcher70] [Goldfarb70] [Shanno70] ont proposé une méthode, dite *BFGS*, permettant d'obtenir H^{k+1} à partir de H^k:

$$H^{k+1} = H^k + \left[1 + \frac{\gamma_k^T H^k \gamma_k}{\delta_k^T \gamma_k}\right] \cdot \frac{\delta_k \delta_k^T}{\delta_k^T \gamma_k} - \frac{\delta_k \gamma_k^T H^k + H^k \gamma_k \delta_k^T}{\delta_k^T \gamma_k} \tag{I.22}$$

$$\text{avec} \begin{cases} \gamma_k = \nabla f(x^{k+1}) - \nabla f(x^k) \\ \delta_k = x^{k+1} - x^k \end{cases}$$

On peut alors décrire brièvement l'algorithme comme suit :

1- Choisir un point de départ x^0 et une matrice initiale H^0 définie positive, par exemple la matrice identité.

2- A chaque itération k :

- On cherche α^k solution du problème unidimensionnel $\underset{\alpha}{Min}\left\{f(x^k - \alpha H^k \nabla f(x^k))\right\}$
- Le nouvel itéré devient $x^{k+1} = x^k - \alpha^k H^k \nabla f(x^k)$
- On met à jour l'approximation de l'inverse du hessien à l'aide de la formule *BFGS* (I.22)

3- Arrêt si convergence sinon retour en **2**

Il existe bien d'autres formules de réactualisation mais la formule *BFGS* est considérée comme la plus performante [Minoux83]. La principale raison est sa relative insensibilité aux imprécisions qui pourraient apparaître lors de la résolution du problème unidimensionnel, ce qui autorise l'utilisation de procédures d'optimisation unidimensionnelles économiques - quoique moins précises - lors de l 'étape 2.

On peut remarquer que si $\delta_k^T \gamma_k > 0$, la matrice H^k est définie positive. [Fletcher70] a montré qu'il était indispensable de procéder à des réinitialisations périodiques des approximations, par exemple à l'aide de la matrice identité, pour assurer le caractère défini positif de H^k tout au long du processus. Ce

procédé permet alors de garantir la convergence globale de l'algorithme. Ici le terme de convergence globale ne signifie pas que l'algorithme converge vers l'optimum global : il indique seulement que, dans le cas bien particulier d'une fonction convexe, l'optimum global sera obtenu. Pour les fonctions quelconques le caractère défini positif de H^k assure que le point stationnaire - $\nabla f(x)=\vec{0}$ -, vers lequel converge l'algorithme, correspond bien à un optimum local [Minoux83].

On peut maintenant utiliser un algorithme d'optimisation complet qui s'appuie sur les méthodes exposées plus haut. Les fonctions objectifs et de contraintes f, (g_i), (h_i) doivent être continûment différentiables. L'optimum global n'est de plus assuré que si les fonctions sont convexes. Dans les autres cas, nous n'aurons a priori qu'un optimum local.

4 METHODES STOCHASTIQUES

4.1 Algorithmes Génétiques (AG)

4.1.1 Introduction

Les algorithmes génétiques ont été créés par J.H. Holland pour mimer les processus observés dans l'évolution des espèces. De nos jours ces processus ne sont pas encore complètement connus mais les biologistes s'accordent sur certains points :

- L'évolution des espèces est un processus qui opère sur des structures appelées chromosomes. Ces structures représentent sous une forme codée les caractéristiques d'un individu (c'est-à-dire un être vivant).

- La sélection naturelle est un processus par lequel les individus qui se sont le mieux adaptés à l'environnement qui les entoure se voient donner une plus grande chance de survie. Les chromosomes de ces individus auront donc une probabilité plus grande de figurer dans les descendances futures.

- Le matériel chromosomique des descendants n'est pas identique à ceux des deux parents. Les chromosomes créés dans les descendances proviennent de la recombinaison ou de la modification des chromosomes hérités des deux parents. Les opérateurs biologiques assurant ce mélange chromosomique se nomment respectivement Croisement (Crossover *en anglais*) et Mutation.

- Le seul arbitre de la sélection est l'adaptation à l'environnement. L'évolution des espèces ne possède pas de mémoire : la seule connaissance sur la manière de produire les individus les mieux adaptés est contenue dans l'ensemble des gènes formant le chromosome.

Ces principes ont intrigué des chercheurs dans les années 60 et les premiers ouvrages sur des algorithmes génétiques sont dus à [Bagley67] ou [Rosenberg67]. Mais ce n'est qu'en 1975 que J.H. Holland [Holland75] jette les bases théoriques d'un algorithme d'optimisation s'appuyant sur des techniques dérivées de la génétique et des mécanismes de la sélection naturelle.

Dans cette partie nous présenterons la théorie de Holland ainsi qu'un algorithme génétique simple. Nous discuterons aussi sur les nombreuses modifications apportées, depuis l'ouvrage de Holland, aux algorithmes génétiques et à leurs opérateurs.

Pour transposer les processus observés dans l'évolution des espèces au domaine de l'optimisation, Holland a introduit deux points fondamentaux :

- D'abord, les structures manipulées par l'algorithme génétique doivent être des chromosomes. Par conséquent, il est nécessaire de transformer l'espace de recherche de notre problème

d'optimisation en un ensemble de chaînes d'un alphabet donné. L'alphabet binaire est le plus couramment utilisé mais n'est pas forcément le plus judicieux [Michalewicz94]. Une chaîne -ou chromosome- correspond donc, sous une forme codée, à une solution potentielle de notre problème.

- Dans la nature, l'adaptation d'un individu reflète sa capacité à survivre dans l'environnement qui l'entoure. En optimisation, l'évaluation de la fonction objectif jouera le rôle de l'environnement. Un individu sera donc d'autant mieux adapté qu'il satisfera bien le critère (de l'optimisation) et c'est cette unique information sur l'évaluation, qui guidera l'AG vers les individus les plus performants. Dans la suite, on appellera adaptation la fonction que l'on cherche à optimiser et on la supposera positive sur tout l'espace de solutions. On fait cette hypothèse car l'adaptation doit traduire une mesure de la performance d'un individu vis à vis de notre problème. Comme en plus cette performance doit être améliorée, on exprimera le problème sous la forme d'une maximisation :

  $$Max\, f(x)$$
  $$avec\ f(x) \geq 0$$

  Nous verrons au paragraphe *4.1.4* comment modifier la fonction objectif pour se ramener à ce type de problème de maximisation.

A partir de ces deux concepts, codage du problème et mesure de l'adaptation d'un individu, on peut dresser le fonctionnement général des AG (Fig. I.2).

```
Program génétique simple

  /* Commentaires */

t          : génération courante
NBGEN      : nombre maximum de générations

P(t), P'(t), P(t+1) sont des populations de même taille N.

  /* Fin commentaires */

début
  t = 0
  initialiser (P(t))
  évaluer (P(t))
  tant que ( t < NBGEN ) faire
    P'(t)=sélection(P(t))
    P(t+1)=croisement&mutation(P'(t))
    évaluer (P(t+1))
    t = t + 1
  fintq
fin
```

Figure I.2: Algorithme génétique simple

L'algorithme génétique opère sur une population d'individus durant plusieurs générations. Chaque individu est en fait constitué d'un seul chromosome. Cela peut surprendre puisque dans la nature les êtres vivants sont en général constitués d'un ensemble de chromosomes (23 paires chez l'Homme). En optimisation génétique, ces cas-là sont très peu répandus [Greene94] [Goldberg87] et dans toute la suite on considérera un individu toujours formé d'un seul chromosome. L'exécution de l'AG débute par l'initialisation aléatoire d'une population d'individus. Ensuite 3 étapes permettent de passer d'une génération à l'autre.

1/ Les chromosomes sont décodés et évalués. Il leur est attribué une probabilité de contribuer à la génération future qui est directement liée à leur adaptation. Ainsi, les individus d'adaptation élevée auront une probabilité plus grande d'avoir des descendants que les individus moins bien adaptés.

2/ L'opérateur de sélection se base sur ces probabilités pour créer une nouvelle population où les individus les mieux adaptés possèdent un nombre plus important de descendants (ou copies). Cet opérateur est une version artificielle de la sélection naturelle.

3/ Enfin, certains individus de cette population sont modifiés par les opérateurs génétiques : l'opérateur de croisement échange des caractéristiques, de façon aléatoire, entre deux individus et l'opérateur de mutation altère certains gènes d'un individu.

Nous allons définir plus en détail le codage des paramètres ainsi que les opérateurs de sélection, de croisement et de mutation introduit par Holland.

4.1.2 L'Algorithme génétique de Holland [Holland75]

A Représentation des paramètres

Les algorithmes génétiques requièrent une représentation sous forme de chromosomes (ou chaînes) des solutions potentielles de notre problème. Holland et De Jong [DeJong75] ont imposé le codage binaire c'est-à-dire qu'un chromosome s'écrit sous la forme d'une chaîne de bits de l'alphabet binaire $A = \{0,1\}$.

Une chaîne $x : b_1 b_2 \cdots b_l$ est décodée en une valeur entière $\sum_{i=1}^{i=l} b_i 2^{i-1}$ où $b_i \in \{0,1\}$.

Pour un paramètre réel x appartenant à l'intervalle $[x^{min}, x^{max}]$, on applique une simple *règle de trois* pour passer de la valeur entière à la valeur réelle. Ainsi, si on considère les chaînes binaires de longueur l on aura les correspondances suivantes (Fig. I.3) :

Chaîne codée		Valeur entière		Valeur réelle
$000\ldots0$	$\leftrightarrow$	0	$\leftrightarrow$	x^{min}
$111\ldots1$	$\leftrightarrow$	2^l-1	$\leftrightarrow$	x^{max}
$b_1 b_2 \cdots b_l$	$\leftrightarrow$	$\sum_{i=1}^{i=l} b_i 2^{i-1}$	$\leftrightarrow$	$x = x^{min} + \left(x^{max} - x^{min}\right)\frac{\sum_{i=1}^{i=l} b_i 2^{i-1}}{2^l-1}$

Figure I.3: Codage binaire des variables

Dans le cas d'un espace de *n paramètres réels*, chaque paramètre $(x_i)_{i=1,n}$ est codé sous la forme d'une chaîne de longueur $(l_i)_{i=1,n}$ puis celles-ci sont concaténées pour former un individu de longueur l.

Bien sûr la longueur d'une chaîne est liée à la précision désirée sur les paramètres. Si *prec* désigne le nombre significatif de chiffres après la virgule souhaité pour le paramètre x_i alors nécessairement :

$$2^{l_i} \geq (x_i^{max} - x_i^{max})10^{prec}$$

Le membre de gauche représente le nombre de chaînes différentes de longueur l_i et le membre de droite le nombre de réels compris dans $\left[x_i^{min}, x_i^{max}\right]$ qu'il faudrait pour satisfaire la précision.

Dans la suite de ce chapitre on ne considérera que des chromosomes binaires.

B Opérateur de sélection

L'opérateur de sélection mime le processus de la sélection naturelle c'est-à-dire que les individus les mieux adaptés ont tendance à se reproduire plus fréquemment. L'opération de sélection se divise en général en deux étapes. La première consiste à attribuer à chaque individu une probabilité d'avoir un descendant dans la génération suivante. Une mise en œuvre simple est de calculer ces probabilités comme étant le rapport de l'adaptation de l'individu sur la somme des adaptations de tous les individus de la population.
Ainsi, pour un individu $\boldsymbol{x}$ d'évaluation $f(\boldsymbol{x})$, la probabilité d'avoir un descendant est :

$$p(\boldsymbol{x},t) = \frac{f(\boldsymbol{x})}{\sum_{k=1}^{N} f(\boldsymbol{x}^k)} \qquad \text{(I.23)}$$

où N est la taille de la population, $(\boldsymbol{x}^k)_{k=1,N}$ l'ensemble des individus et t la génération courante.
Cette sélection est nommée *sélection proportionnelle* puisque la probabilité d'avoir un descendant est proportionnelle à la valeur de l'adaptation de l'individu.

On pose aussi $tsr(\boldsymbol{x},t)$ le nombre de descendants attendus, lié simplement à $p(\boldsymbol{x},t)$ par la relation $tsr(\boldsymbol{x},t) = N\,p(\boldsymbol{x},t)$.

La seconde étape consiste à se baser sur ces probabilités pour former une population de même taille à l'aide de copies -ou descendants- des individus de la population appelée à se reproduire. Pour cela, on crée une roue de loterie biaisée pour laquelle chaque individu occupe une section proportionnelle à $p(x,t)$. La roue est activée de manière aléatoire un nombre de fois égal à la taille de la population et, à chaque coup de roue, l'individu désigné par l'aiguille est copié dans la nouvelle population. Ce n'est en fait rien d'autre qu'un algorithme d'échantillonnage : on convertit des valeurs réelles (les probabilités) en valeurs entières (copies).

C Opérateurs de Croisement et Mutation

Ces deux opérateurs agissent sur les chromosomes sélectionnés précédemment par l'opérateur de sélection. Contrairement à ce dernier qui favorise la propagation des meilleures chaînes, les opérateurs de croisement et de mutation s'occupent de promouvoir l'exploration de nouvelles régions de l'espace de recherche.

Opérateur de croisement

Le croisement est un processus aléatoire qui se charge d'échanger une partie des matériels génétiques de deux chromosomes parents pour créer deux enfants. En pratique, l'échange n'est effectué que si une probabilité ***pc*** est "passée". Dans ce cas un point de coupe est choisi aléatoirement le long des chaînes et les bits situés à droite de ce point sont échangés entre les deux parents (Fig. I.4).

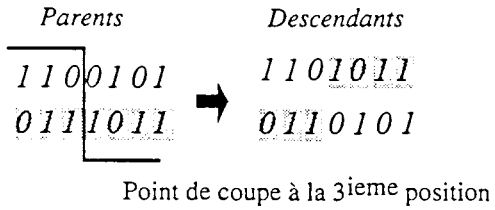


Figure I.4: Croisement de 2 individus

Opérateurs de mutation

La mutation est aussi un processus aléatoire qui se charge d'introduire des variations dans une chaîne. Typiquement une chaîne est choisie aléatoirement dans la population et ses bits sont inversés si une probabilité ***pm*** est passée (Fig. I.5).

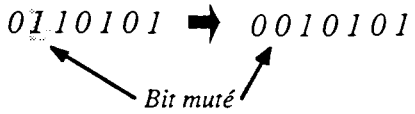


Figure I.5: Mutation d'un individu

Intérêt de ces opérateurs

L'opérateur de croisement a pour effet de combiner des informations provenant de deux chromosomes. Si ces chromosomes contiennent des informations génétiques importantes sur ce qui est bon pour notre problème alors leur combinaison peut engendrer un chromosome encore plus performant. Pour cette raison il est utilisé avec un taux *pc* assez élevé.

En pratique, lorsque les deux parents possèdent un patrimoine génétique très différent, le croisement permet l'exploration de nouvelles zones de l'espace de recherche. L'action de la sélection, qui pousse les meilleures structures à se reproduire très fréquemment, réduira les chances de recombinaison d'individus génétiquement différents. Cette perte de diversité dans la population peut alors déboucher sur une convergence de la population vers une région de l'espace qui ne contient qu'un optimum local. Pour cette raison l'opérateur de mutation doit être utilisé car il permet de relancer l'exploration vers des régions qui n'ont pas pu être atteintes par le simple effet du croisement. Il assure ainsi que la probabilité d'atteindre un point quelconque dans l'espace de recherche n'est jamais nulle. Un algorithme génétique demande donc un équilibre entre l'exploration de l'espace de recherche et l'exploitation des meilleures structures. De cet équilibre provient sa capacité à localiser l'optimum sans se faire piéger dans les optima locaux. C'est ce qui le différencie d'autres algorithmes stochastiques comme le Hill Climbing [Pogu92] qui ne cherche qu'à améliorer la solution courante ou les algorithmes purement aléatoires qui explorent de façon exhaustive l'espace des solutions en négligeant d'exploiter les régions les plus prometteuses. Pour ces raisons les AG ont été appliqués dans de nombreux domaines avec succès [Vignaux91] [Shahookar90] [Smith92] [Kosigo94] [Nolan94].

Ci dessous sont résumées les principales caractéristiques d'un algorithme génétique.

1. Les algorithmes génétiques travaillent sur un **codage des paramètres**.

2. Les algorithmes génétiques utilisent une **population de points** de l'espace de recherche et non un point unique.

3. La seule information disponible est l'évaluation de la fonction. Aucune autre information comme le gradient n'est nécessaire. L'AG s'applique donc à une large gamme de problèmes.

4. Les **règles** d'exploration de l'espace de recherche sont **probabilistes** et non pas déterministes.

D Théorie des schémas

Les AG utilisent comme seules informations les N valeurs d'adaptations des individus. En fait, les similarités génétiques entre les chaînes apportent une information supplémentaire pour guider l'exploration. L'étude de ces similarités a été effectuée par Holland et porte le nom de *théorie des schémas*.

Définition : On considère l'alphabet binaire augmenté $A^+=\{0,1,*\}$ où le symbole "*" représente un joker qui peut prendre indifféremment les valeurs "0" ou "1". On appelle **schéma** H de longueur l une chaîne $H = a_1 a_2 \cdots a_l$ où $\forall i \in [1,l]$ $a_i \in A^+$.

Ainsi, H représente un ensemble de chaînes binaires qui ont certaines caractéristiques génétiques communes.

Par exemple le schéma H=1*000* représente les 4 chaînes {100000, 110000, 100001, 110001}. Dans une population de taille N, on dénombre entre 2^l et $Nx\,2^l$ schémas [Holland75].

Définition : L'**ordre** d'un schéma est le nombre de positions fixes de H c'est-à-dire le nombre de "0" ou de "1". Il se note o(H).

H = 1*000* o(H) = 4

Définition : La **longueur fondamentale** δ(H) d'un schéma H est la distance séparant la première position fixe de la dernière.

H = 1*000* δ(H) = 5-1 = 4

Les schémas ne sont pas manipulés directement par les AG. Ils permettent seulement de rendre compte des similarités entre les individus pour expliquer de façon théorique les performances des AG.

Effets de la Reproduction sur les schémas

Soit m(H,t) le nombre de schémas H à la génération t. Soit $\{A_1, A_2, \cdots, A_N\}$ la population de taille N à cette génération.

Par sélection les individus A_i se reproduisent avec une probabilité $p(A_i)=\dfrac{f(A_i)}{\sum_{i=1}^{i=N} f(A_i)}$ dans la génération suivante.

Statistiquement on aura donc $m(H,t+1)= \sum_{A_i \in S(H)} N \times p(A_i)$ où S(H) est l'ensemble des individus représentés par le schéma H.

$$m(H,t+1) = \sum_{A_i \in S(H)} \left(N \times \frac{f(A_i)}{\sum_{j=1}^{j=N} f(A_j)} \right) = \frac{N}{\sum_{j=1}^{j=N} f(A_j)} \times \sum_{A_i \in S(H)} f(A_i) \qquad (I.24)$$

En posant $\bar{f}(t) = \frac{\sum_{i=1}^{i=N} f(A_i)}{N}$ l'adaptation moyenne de la population et

$\bar{f}(H) = \frac{\sum_{A_i \in S(H)} f(A_i)}{m(H,t)}$ l'adaption moyenne du schéma H

on a $m(H,t+1) = m(H,t) \times \frac{\bar{f}(H)}{\bar{f}(t)}$ (I.25)

Pour un schéma H d'adaptation supérieure à la moyenne on peut poser $\bar{f}(H) = (1+C)\bar{f}(t)$ avec $C > 0$ on a alors $m(H,t) = (1+C)^t\, m(H,0)$ en supposant C constant dans le temps. La reproduction a donc pour effet d'augmenter de façon exponentielle la part des schémas performants dans les générations successives. Quant aux schémas de faibles adaptations, ils sont condamnés à dépérir.

Effet du Croisement sur les schémas

Si le croisement s'effectue entre la première et la dernière position fixe du schéma alors celui-ci risque d'être détruit. On peut donc majorer la probabilité de destruction par $pc\frac{\delta(H)}{l-1}$ où pc est la probabilité de croisement. On en déduit une probabilité de survie ps pour le schéma H : $ps \geq 1 - pc\frac{\delta(H)}{l-1}$

L'effet combiné de la reproduction et du croisement nous donne alors :

$$m(H,t+1) \geq m(H,t) \times \frac{\bar{f}(H)}{\bar{f}(t)} \left(1 - pc\frac{\delta(H)}{l-1} \right) \qquad (I.26)$$

Les schémas bien adaptés et de longueur fondamentale $\delta(H)$ courte vont donc augmenter de façon exponentielle dans les générations successives. Ce résultat porte le nom de **Théorème des Schémas.**

Effet de la mutation sur les schémas

Soit pm la probabilité d'inversion d'un bit dans un individu. Un schéma survivra si toutes ses positions fixes ne sont pas détruites. La probabilité de survie d'un schéma d'ordre o(H) est donc $(1-pm)^{o(H)}$. En introduisant cette probabilité dans l'équation (I.26), on obtient :

$$m(H,t+1) \geq m(H,t) \times \frac{\bar{f}(H)}{\bar{f}(t)} \left(1 - pc \frac{\delta(H)}{l-1} \right) (1-pm)^{o(H)} \qquad (I.27)$$

Les schémas courts, d'ordre peu élevé et d'adaptation supérieure à la moyenne augmentent donc de façon exponentielle dans les générations successives. On nomme de tels schémas blocs élémentaires et ils constituent l'information supplémentaire, manipulée indirectement par l'algorithme génétique. Holland a estimé, qu'à chaque génération, dans une population de N individus, environ N^3 blocs élémentaires sont traités par l'AG. Ainsi, sans utiliser d'autre mémoire que celle constituée par la population elle-même, environ N^3 informations utiles sont manipulées : c'est le *parallélisme implicite* de Holland. L'AG explorerait donc l'espace de recherche par la concaténation de schémas de longueurs fondamentales faibles et de grandes performances. Cependant, cette hypothèse n'a pu être prouvée [Ankenbrandt91]. Elle s'apparente plus à un acte de foi et a pu être mise en défaut dans certains problèmes dits déceptifs [Goldberg89a].
L'hypothèse des blocs élémentaires justifie aussi l'emploi d'un alphabet de faible cardinalité comme l'alphabet binaire. En effet dans ce cas le nombre de schémas disponibles pour l'AG est maximisé [Goldberg89b].

L'étude des schémas par Holland permet donc d'avoir une idée sur la manière dont les AG explorent l'espace de recherche pour converger vers l'optimum global. Mais en pratique il arrive que cette convergence survienne trop tôt et que l'AG reste bloqué sur un optimum local [Booker87]. Cette convergence prématurée résulte souvent d'un déséquilibre entre l'exploitation des meilleurs et le maintien d'une certaine diversité dans la population [Whitley89]. Comment garantir à coup sûr cet équilibre et combien de temps doît il être maintenu avant que la population ne converge (enfin !) vers l'optimum (que l'on espère être global !) ?
En fait, il semble qu'il n'y ait pas de solutions infaillibles à cette question. Cependant, de nombreux auteurs se sont efforcés de "combattre" les phénomènes de convergence prématurée en implantant, par exemple, de nouveaux opérateurs ou en modifiant ceux vus plus haut. Dans ce qui suit, nous allons voir plus en détails les problèmes qui peuvent survenir à chaque étape de l'AG de Holland (Sélection-Croisement-Mutation). Nous présenterons aussi les solutions qui peuvent être mises en œuvre.

4.1.3 Modifications sur les opérateurs

A Sélection

On divise la phase de sélection en deux étapes [Grefenstette89]. Dans la première on associe à chaque individu x un nombre réel $tsr(x,t)$ qui représente le nombre de descendants attendus pour l'individu x à la génération t. Dans la seconde, un algorithme d'échantillonnage convertit ce nombre $tsr(x,t)$ en un nombre entier qui constitue le nombre de copies de l'individu x dans la nouvelle génération. Holland a

utilisé une méthode de sélection, dite sélection proportionnelle , où le nombre de descendants attendus est directement proportionnel à la fonction d'adaptation de l'individu c'est-à-dire :

$tsr(x,t)=\frac{f(x)}{\bar{f}(t)}$ avec $\bar{f}(t)$ l'adaptation moyenne.

L'emploi d'une telle sélection se heurte à divers types de problèmes. Par exemple, en début d'exécution (ou en cours !) la population peut contenir un individu extraordinaire (du point de vue de sa performance !). Cet individu va donc se voir attribuer une place importante dans les générations successives ce qui peut résulter dans la convergence prématurée de l'AG vers une solution non optimale puisqu'il existera alors peu de chances de recombinaison entre individus différents. De la même façon, deux schémas moyens ou médiocres lorsqu'ils sont considérés séparément mais qui deviennent très performants lorsqu'ils sont recombinés, risquent de disparaître prématurément sans s'être jamais recombinés. Une pression moins forte en direction des meilleurs individus aurait permis de conserver ces deux schémas plus longtemps et leur aurait laissé une chance de se recombiner. Un autre problème avec ce type de sélection se produit en fin d'exécution. En effet, la population ayant convergé, la différence entre l'adaptation des différents individus devient moins importante. La sélection proportionnelle va donc attribuer sensiblement le même nombre de descendants à chaque individu ce qui entraîne une stagnation de la recherche. Pour relancer l'AG il aurait fallu marquer plus fortement les différences entre les adaptations des individus. Le terme de **pression de sélection** est souvent employé pour rendre compte des effets pervers de l'opérateur de sélection. Une pression de sélection trop élevée a pour effet de trop diriger la recherche vers les meilleurs (à la manière d'un gradient !) et se fait donc au détriment de la diversité dans la population. Ce déséquilibre peut entraîner, comme dans le cas de la sélection proportionnelle, une convergence de l'AG vers une solution non optimale. A contrario, une pression de sélection faible permet d'augmenter la capacité d'exploration de l'AG en impliquant plus d'individus différents (et donc plus de schémas) dans la recherche. Mais si cette pression vers les meilleurs est trop faible, la recherche risque d'être inefficace et conduire à des phénomènes de stagnation.

Diverses méthodes ont été proposées pour contrôler la pression de sélection au cours de l'exécution. **Certaines conservent la sélection proportionnelle** mais transforment la fonction d'adaptation (dite "brute") avant de calculer les taux *tsr(x,t)*. Dans cette catégorie on trouvera :

1/ *Le Fenêtrage* : Dans cette méthode, proposée par [Grefenstette86] , l'adaptation transformée f_{trans} se calcule simplement à partir de l'adaptation f par la relation $f_{trans} = f - f_{min}$.
où f_{min} est l'adaptation minimum qui est réactualisée à l'aide d'un paramètre de fenêtrage W. Cette méthode permet surtout d'éliminer les problèmes de stagnation en fin d'exécution. Par exemple, pour une population dont les adaptations sont comprises dans l'intervalle [105,110], un choix de f_{min}=100 permet d'éviter une stagnation. En effet, en ramenant les adaptations dans l'intervalle [5,10] le meilleur

individu possédera statistiquement 2 fois plus de chances de posséder un descendant que le moins brillant.

2/ *Transformation linéaire.* Dans cette méthode l'adaptation est modifiée linéairement $f_{trans}=af+b$ [Goldberg89a]. Les paramètres a et b sont calculés à chaque génération de sorte que :

1- La moyenne de l'adaptation transformée soit égale à la moyenne de l'adaptation brute.
2- L'adaptation transformée du meilleur individu soit égale à un multiple C de la moyenne de l'adaptation brute.

Ainsi, par sélection proportionnelle, on s'assure que les individus d'adaptation moyenne auront un descendant et que le meilleur individu en aura au plus C. En prenant pour C des valeurs comprises entre 1,2 et 2 on pourra donc éviter les problèmes de convergence prématurée. Selon [Goldberg89a], cette méthode se révèle efficace. Par contre, en fin d'exécution, si un individu de très mauvaise adaptation ($<<f_{moy}$) apparaît dans la population alors le mécanisme de transformation introduit des valeurs d'adaptations négatives qui posent problème.

3/ *Troncature Sigma.* Cette méthode [Forrest85a] se présente comme une amélioration de la transformation linéaire [Michalewicz94].

$$f_{trans} = g(f-(\bar{f}-C\sigma))$$

où $$g(x)=\begin{cases} x \ pour \ x>0 \\ 0 \ sinon \end{cases}$$

C est entier, choisit aléatoirement dans l'intervalle [1,5] et σ représente l'écart type de l'adaptation dans la population.

4/ *Transformation Puissance* [Gillies85] suggère la transformation $f_{trans}=f^k$ et propose le choix de k=1,005 en précisant qu'en fait, la valeur de k est très liée au problème étudié.

5/ *Transformation de Boltzmann* [DeLaMaza93]

$$f_{trans} = e^{f/T}$$

Dans cette transformation T joue un rôle similaire à celui de la température dans l'Algorithme de Recuit Simulé (que nous verrons plus loin). Il permet de contrôler directement la pression de sélection: quand T est élevé la pression de sélection est faible et inversement.

D'autres mécanismes de sélection ont été proposés. Ils se différencient des précédents car il ne mettent plus en jeu la sélection proportionnelle. Par exemple [Baker85] calcule le nombre de descendants attendus, de façon linéaire, en fonction du rang de l'individu dans la population. Pour cela

il utilise un paramètre MAX qui désigne le nombre de descendants pour le meilleur individu et un paramètre MIN qui désigne le nombre de descendants pour le moins bon. Enfin, il projette le nombre *tsr(x,t)* entre [MIN,MAX] à l'aide de la formule :

$$tsr(x,t) = \text{MIN} + (\text{MAX-MIN}) \frac{rank(x,t)-1}{N-1}$$

où *rank(x,t)* désigne le rang de l'individu x à la génération t dans la population. On prendra *rank* =1 pour le moins bon et *rank* = N pour le meilleur. Le nombre de descendants attendus étant un nombre positif et la somme $\sum tsr(x,t)$ étant égale à N, on a nécessairement MAX $\in$ [0,2] et MIN=2-MAX. La pression de sélection peut donc être contrôlée par l'intermédiaire du paramètre MAX.

[Michalewicz 94] propose une interpolation similaire mais non linéaire:

$$tsr(x,t) = \text{N x q x } q^{rank-1} \text{ avec } q \in [0,1]$$

Par rapport aux méthodes précédentes ces approches évitent d'avoir recours aux diverses transformations d'échelle sur l'adaptation pour gérer la pression de sélection. C'est d'ailleurs le rang qui introduit de façon automatique une échelle dans la population. De plus, la pression de sélection est ici contrôlée très simplement grâce aux paramètres MAX ou q et de façon plus précise [Whitley89]. [Brindle81] a utilisé une sélection moins classique : **le tournoi stochastique**. Cette sélection ne calcule aucun taux et ne nécessite donc pas d'algorithme d'échantillonnage. L'idée est simple : il suffit de choisir aléatoirement k individus et de sélectionner le meilleur pour le placer dans la nouvelle génération. Ensuite, on recommence jusqu'à ce que la population soit complète. En général ce nombre k n'est pas choisi trop grand [Goldberg91].

Dans la littérature on ne trouve pas de comparaisons suffisantes entre ces diverses méthodes de sélection pour tirer des conclusions définitives. Néanmoins, [DeLaMaza93] et [Grefenstette89] ont mené des études théoriques qualitatives pour montrer l'influence des diverses transformations de l'adaptation sur la pression de sélection. Mais ces études n'offrent pas de réelles comparaisons sur leurs performances respectives. [Golberg91] compare la sélection proportionnelle sans transformation, la sélection de [Baker85] et le tournoi de [Brindle81] de manière plus quantitative. Pour cela, il a simulé le comportement d'une population contenant déjà l'optimum et évoluant sous la conduite de la seule sélection. Une des mesures portait sur le taux de croissance du meilleur individu entre deux générations successives. Les calculs théoriques ont montré qu'une sélection proportionnelle classique possède un taux de croissance très élevé dans les premières générations (convergence prématurée) mais qui devient faible dès lors que le meilleur individu occupe 50% de la population (stagnation !). A contrario les deux autres méthodes maintiennent un taux de croissance compris entre 1 et 2 tout au long de l'exécution. Une deuxième mesure portait sur le temps de convergence c'est-à-dire la génération pour laquelle le meilleur individu occupe 90% de la population. Dans le cas d'une sélection

proportionnelle le temps est en $0(Nlog(N))$ et dans les deux autres cas en $0(log(N))$. Attention, ces temps ne reflètent pas la réalité d'un AG puisqu'il considère que le meilleur individu a déjà été formé et ils ne tiennent pas compte des effets des opérateurs de croisement et de mutation. Cependant, ils peuvent donner une idée sur le temps qu'un AG doit tourner avant que la mutation ne devienne le principal mécanisme chargé de l'exploration. Pour terminer, Goldberg a calculé la complexité de ces 3 méthodes. Elle est de l'ordre de $0(N)$ pour le tournoi et, suivant l'algorithme d'échantillonnage utilisé, elle varie entre $0(N)$ et $0(N^2)$ pour la sélection proportionnelle et entre $0(Nlog(N))$ et $0(N^2)$ pour la sélection par rang. Goldberg préconise l'utilisation du tournoi pour sa mise en parallèle plus aisée mais dans ce cas le contrôle de la pression de sélection est plus difficile à apprécier.
D'une façon générale, toutes les méthodes présentées peuvent conduire à des résultats sensiblement équivalents sur un même problème pourvu que les paramètres de la sélection soient convenablement choisis [DeLaMaza93]. Mais, il semble que la sélection par rang soit la plus simple à mettre en œuvre et finalement celle qui permet de gérer le mieux la pression de sélection. Ceci a été confirmé par les tests de [Bäck91a].

B Algorithmes d'échantillonnage

La section précédente présentait diverses méthodes de calcul du nombre de descendants attendus dans le but de contrôler au mieux les effets de la pression de sélection. La seconde étape de la sélection consiste à convertir ces nombres réels en valeurs entières à l'aide d'un algorithme d'échantillonnage. L'algorithme présenté par Holland simulait une roue de loterie biaisée. Cependant, ce type d'algorithme peut entraîner des erreurs puisque le nombre de descendants réellement attribués à un individu peut être significativement différent du nombre de descendants attendus. En effet, il suffit d'imaginer que la roue tombe N fois sur le même individu (cas extrême !). Il est sûr que ces erreurs apportent un peu de diversité en réintroduisant, par exemple, des individus qui auraient dû être éliminés. Mais ce rôle est plutôt dévolu aux opérateurs de Croisement et Mutation. Pour réduire ces erreurs d'échantillonnage [Brindle81] a proposé un algorithme dit *déterministe pour la partie entière et stochastique pour la partie restante*. Dans cet algorithme, le nombre de copies allouées à l'individu est au moins égal à la partie entière de $tsr(x,t)$. Ensuite, les parties décimales sont placées sur une Roue de Loterie biaisée pour affecter les dernières places dans la population.

[Baker87] a présenté un modèle de *Roue généralisée*. Dans ce cas N aiguilles sont réparties uniformément autour de la Roue et en un tirage, les individus contribuant à la nouvelle population sont désignés. [Baker87] a étudié de façon empirique les risques d'erreurs qui pouvaient être engendrés par divers algorithmes d'échantillonnage. Il préconise l'utilisation de sa Roue généralisée ou de l'Algorithme de Brindle, qui présente en plus, la caractéristique d'être facilement parallélisable.

C Croisement

Le crossover 1-point introduit par Holland était inspiré du Crossing Over biologique. Mais sa caractéristique fondamentale est de recombiner les schémas. Malheureusement, si on considère deux individus:

1 1 0 1 1 0 0 1 0 1 1
et 0 0 0 1 0 1 1 0 1 1 1

contenant respectivement les schémas :

1 * 0 * * * * * * * 1
et * * * * 0 * * * 1 * *

alors aucun croisement 1-point ne sera capable de les recombiner. Pour parer à ce type de carences, un croisement possédant deux points de coupe a été introduit. Ainsi, deux positions choisies aléatoirement sont sélectionnées et les matériels génétiques entre ces deux points sont échangés entre les deux individus [DeJong75]. Finalement des croisements n points ont été étudiés [Spears91a].
Syswerda [Syswerda89] a introduit un opérateur de croisement un peu différent. Plutôt que d'échanger des segments de chaîne, il propose d'échanger les bits entre les deux parents avec une probabilité P_O, pour chacun des bits, prise égale à 0,5. Syswerda a montré que la probabilité de survie d'un schéma suite à un croisement uniforme (U.C.) était inférieure à celle dans le cas des croisements 1-point et 2-points. On comprend bien que de tels échanges bit à bit augmentent le risque de détruire un schéma. Mais il s'est aussi intéressé au taux de recombinaison des schémas. Pour des schémas de longueur fondamentale supérieure à 12, l'U.C. possède le meilleur taux de recombinaison. Sur les 6 fonctions tests qu'il a utilisé pour évaluer son U.C., il a obtenu des résultats sensiblement meilleurs.

La nature plus destructrice de cet opérateur de croisement n'est pas toujours une mauvaise chose. Par exemple, en fin d'exécution, lorsque la population devient homogène, un facteur devient important : c'est de savoir si un descendant créé par croisement sera différent de ses parents. En somme, il faut se demander si un croisement sera capable de générer de nouvelles structures à partir de parents possédant un matériel génétique sensiblement identique. De par sa nature, l'U.C. possède justement une certaine capacité à créer de telles nouvelles structures et donc de permettre une exploration même en fin d'exécution. Ceci suggère aussi d'utiliser l'U.C. avec des populations de faibles tailles pour pallier le manque d'information.

Dans le cas de larges populations, la probabilité de destruction de l'U.C. est encore trop élevée et on lui préfère plutôt un croisement 2-points [Spears91b]. En fait, la probabilité de destruction des U.C. peut être diminuée de manière sensible en jouant sur le paramètre P_O [Spears91b]. Ainsi, même avec une large population, la probabilité de destruction d'un schéma d'ordre 3 est approximativement la même

pour le croisement 2-points et le U.C. avec $P_O = 0,2$. Avec l'avantage que la probabilité de destruction des schémas par l'U.C. reste identique quelle que soit la longueur fondamentale du schéma.

D Mutation

La mutation préserve de la perte d'informations et permet l'exploration de régions de l'espace de recherche qui ne sont pas accessibles par le simple croisement. L'opérateur de mutation le plus simple est celui proposé par Holland mais on rencontre aussi l'opérateur de mutation dynamique [Janikow91]. Cet opérateur agit de façon non uniforme sur les bits. Ainsi, en début d'exécution, il inversera plus fréquemment les bits de poids forts permettant de cette manière une exploration de régions très éloignées. Par contre, en fin d'exécution, lorsque la population aura convergé vers l'optimum, la mutation inversera plus fréquemment les bits de poids faibles: elle se comporte alors plus comme un opérateur de voisinage. Une mise en œuvre simple est proposée ci dessous :
Si la k^{ième} variable est mutée dans la chaîne alors elle le sera sur le bit *pos* (*pos*=0 correspond au bit de poids fort)

avec $pos = n\left(1 - r^{(1-t/T)^b}\right)$

où n est le nombre de bits codant la variable k, T le nombre maximum de générations, b un paramètre traduisant la "non uniformité" plus ou moins importante de l'opérateur et enfin r un nombre aléatoire réel compris dans l'intervalle [0,1].

L'importance relative de la mutation par rapport au croisement dans un AG est source de débat. "L'école" de Holland [Holland75] [DeJong75] [Goldberg89] considère que le croisement est l'opérateur le plus important et de loin. Beaucoup d'efforts ont été consentis pour l'analyse du croisement et de son influence sur l'AG [DeJong75] [Spears91a] [Spears91b] [Vose91]. Dans ces analyses, la mutation est souvent considérée comme un opérateur d'importance secondaire. D'un autre côté "l'école" allemande a mené des recherches sur les "stratégies de l'Evolution" où l'opérateur clé est la mutation [Bäck91b]. [Schaffer89] a confirmé et illustré la force de la mutation dans les AG. Pour [Fogel90], la mutation et le croisement ont la même capacité de recherche. A l'heure actuelle la seule comparaison théorique sur l'importance relative des 2 opérateurs a été faite par [Spears95]. Ses résultats donnent une justification théorique au rôle tenu par le croisement vis à vis de la théorie de Holland. C'est bien cet opérateur qui est le plus à même de construire des blocs élémentaires. La mutation ne peut pas tenir ce rôle de façon aussi satisfaisante. Mais faut il croire en la recombinaison d'informations élémentaires par la nature ? [Fogel90] pense plutôt que tout réside dans l'équilibre à trouver entre l'exploration et l'exploitation. Et dans ce cas, il est difficile de définir l'importance relative des deux opérateurs quant à leurs effets sur la diversité.

Dans les AG classiques la mutation est appliquée avec un taux faible. En effet si cet opérateur est trop présent, l'AG se ramène à une simple recherche aléatoire. [DeJong75] a proposé, pour des populations de tailles 50 à 100, des probabilités pc=0,6 et pm=0,01. [Grefenstette86], pour une population de 80 individus, conservait des taux pc=0,95 et pm=0,01. Ces paramètres ont été validés ensuite par [Shaffer89]. [Bäck93] a calculé un taux de mutation optimal de façon empirique. Ce taux est égal à $1/l$ où l est la longueur de la chaîne. Cependant, il s'agissait, non pas d'un codage binaire classique mais d'un codage de *Gray*. L'intérêt d'un tel codage est qu'il permet de projeter des voisins euclidiens en voisins de Hamming c'est-à-dire que des entiers adjacents seront représentés par des chaînes qui ne diffèrent que d'un bit. La table I.1 présente le code de Gray sur 4 bits.
Cette représentation de Gray se révèle en général plus performante que la représentation classique [Schaffer89].

Valeur entière	Binaire	Gray
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Table I.1: Code de Gray

Dans ce qui précède, nous avons mis en évidence les problèmes qui peuvent survenir au cours du processus d'optimisation génétique. Ces problèmes sont, pour la plupart, dus à des déséquilibres entre l'exploration de l'espace des solutions et l'exploitation des meilleures structures. Bien sûr, l'algorithme doit converger, mais cette convergence ne doit pas se faire trop rapidement, sinon on court le risque de passer à côté de l'optimum global. Les solutions présentées constituent en fait une aide au réglage et au choix des bons paramètres de l'algorithme.

4.1.4 Adaptation et fonction objectif

L'adaptation d'un individu traduit une mesure de performance. L'objectif de notre optimisation devra donc être exprimé en termes de maximisation. De plus, le nombre de descendants est directement lié à la valeur de la fonction donc naturellement celle-ci devra être positive sur tout l'espace de recherche. Pour satisfaire ces deux contraintes il sera souvent nécessaire de réajuster la fonction. Par exemple pour un problème

(P) $Min\, f(x) \in \Re$

on pourra définir une fonction d'adaptation [Golberg89a]:

$Max\ \Phi(x) \in \Re^+$

avec $\Phi(x) = \dfrac{1}{f_{max} + f(x)}$

où f_{max} peut être choisi supérieur au maximum de f observé dans la population courante.

Prise en compte des contraintes

La méthode couramment employée pour traiter les contraintes consiste à les incorporer dans la fonction objectif (cf. 3.2.1). Classiquement, on prendra une fonction de pénalités extérieures [Michalewicz94] :

Un problème (P) $\begin{cases} Min\, f(\boldsymbol{x}) \in \Re \\ g_i(\boldsymbol{x}) \le 0 \qquad i = 1,\ldots,m \\ h_j(\boldsymbol{x}) = 0 \qquad j = 1,\ldots,l \end{cases}$

sera transformé en

$$Max\, \Phi(\boldsymbol{x}) = \frac{1}{f_{max} + f(x) + r\sum\left[max[0, g_i(x)]\right]^2 + r\sum h_i^2(x)}$$

où r représente le facteur de pénalité.

Cependant, l'emploi d'une telle pénalisation, avec une sélection accordant un nombre de descendants proportionnel à la valeur de l'adaptation, peut poser problème. En effet, si le paramètre r est choisi trop grand et si la majorité des individus violent les contraintes alors un individu ne les violant pas dominera rapidement (trop !) la population. De la même manière, si r est trop faible alors les individus qui violent les contraintes risquent de dominer la population. Pour éviter ce type de problème, lié à une perte d'information sur la valeur de l'adaptation *brute*, il est préférable de choisir une sélection basée sur le rang [Powell93].

Powell a choisi une sélection par rang de Baker et a redéfini la fonction d'adaptation comme suit (pour une minimisation):

Si les contraintes sont satisfaites :

$\Phi(x) = S(f(x))$ où S est une fonction qui projette f dans $[-\infty,1]$

Si une ou plusieurs contraintes sont violées alors :

$$\Phi(x) = 1 + r\sum\left[max[0, g_i(x)]\right]^2 + r\sum h_i^2(x)$$

On peut noter que l'adaptation prend des valeurs négatives. En fait, les problèmes de positivité de la fonction étaient surtout dus au choix d'une sélection proportionnelle : $tsr(x,t)$ ne peut être négatif ! Avec une sélection par rang, ces contraintes ne se posent plus.
Finalement, cette projection assure que les points réalisables seront toujours préférés lors du mécanisme de sélection . Dans le même temps, les points violant les contraintes ne sont pas écartés de la sélection.

4.2 Recuit Simulé

4.2.1 Introduction

Le Recuit Simulé [Kirkpatrick83] a été développé par analogie entre un problème d'optimisation et le processus de recuit utilisé en métallurgie pour l'obtention d'un matériau sans impuretés (dans un état de minimum d'énergie). Au début du processus de recuit réel on élève la température du matériau. Le matériau se trouve alors dans un état d'énergie élevée. On le refroidit ensuite très lentement en marquant des paliers de température. A chaque palier, les atomes s'ordonnent de façon à atteindre l'équilibre thermique correspondant à une valeur de l'énergie stable. Cette stratégie d'abaissement de la température permet d'obtenir en fin de processus un matériau dans un état cristallin bien ordonné correspondant à un état d'énergie minimum. Par contre, si la baisse de température se fait de manière trop brutale on obtient un matériau amorphe où les atomes sont figés dans un état désordonné traduisant un minimum local de l'énergie.
Le comportement des atomes a été expliqué au 19^{e} siècle et l'équilibre thermique a été caractérisé grâce à une loi statistique de distribution proposée par Boltzman : pour une température donnée T, la probabilité pour qu'un système d'atomes soit dans un état d'énergie E est proportionnelle à exp(-E/T). Ainsi quand la température décroît et devient proche de zéro, seuls les états d'énergie minimum ont une probabilité non nulle d'apparaître.

4.2.2 Algorithme de Recuit Simulé

Métropolis [Métropolis53] a proposé une méthode simulant l'évolution d'un système d'atomes vers l'équilibre thermique. Connaissant l'état courant du système, on explore une configuration microscopique voisine et la différence d'énergie ΔE entre la configuration courante et la configuration perturbée est calculée.

Si $\Delta E<0$ alors la nouvelle configuration est sauvegardée et est utilisée comme solution de départ d'une nouvelle perturbation.

Si $\Delta E>0$ alors on conservera la nouvelle configuration avec une probabilité $P=\exp(-\Delta E/k_B T)$. On peut noter que plus la température est élevée et plus la nouvelle configuration a des chances d'être acceptée.

[Kirkpatrick83] a transposé cette méthode d'évolution à l'optimisation combinatoire grâce aux correspondances suivantes (Table I.2):

Recuit Réel	Optimisation
Arrangements des atomes	Paramètres du problème
Energie	Objectif à minimiser
Température	Paramètre de contrôle
Minimum de l'Energie	Minimum global

Table I.2: Correpondances Recuit réel-Optimisation

En utilisant ces correspondances, le Recuit Simulé peut être vu comme une succession d'algorithmes de Métropolis pour des valeurs décroissantes de la température.

Dans l'algorithme classique, on débute avec une température initiale élevée. Une configuration initiale -valeurs des paramètres- est choisie au hasard et, à l'aide d'une transformation de voisinage, une nouvelle configuration est générée. Ceci correspond en fait à une "petite" perturbation aléatoire sur les variables. On évalue ensuite notre critère E pour ces deux configurations et on calcule l'écart ΔE. La nouvelle configuration remplace alors la configuration initiale avec une probabilité P :

où $P = 1$ si le critère s'améliore $(\Delta E < 0)$

$P = exp(-E/k_B T)$ sinon $(\Delta E > 0)$

On réitère ce processus, à partir de la configuration courante, un certain nombre de fois jusqu'à obtenir l'équilibre thermique. Ce mécanisme génère en fait une séquence de solutions potentielles à notre problème. Cette séquence peut être décrite par une chaîne de Markov puisque chaque nouvelle solution ne dépend que de la solution précédente.

Ensuite, on abaisse progressivement la température, en générant à chaque palier une séquence de configurations potentielles. L'algorithme (Fig. I.6) s'arrête lorsque la température atteint une certaine valeur.

```
Program recuit simulé

    /* Commentaires */

T0      : Température initiale
Tfin    : Température finale
E       : Fonction objectif

    /* Fin commentaires */
début
    choisir une configuration initiale x0
    k ← 0
    tant que ( Tk > Tfin ) faire
       répéter
          choisir xi+1 dans le voisinage de xi
          calculer ΔE = E(xk+1)-E(xk)
          générer un nombre aléatoire Pr ∈ [0,1]
          Si (exp(-ΔE /Tk) > Pr) alors
             xi+1 ← xi
             i ← i+1
          fsi
       jusqu'à ( l'équilibre )
       Tk+1 ← Abaisser(Tk)
       k ← k+1
    fintq
fin
```

Figure I.6: Algorithme de Recuit Simulé

Contrairement à d'autres algorithmes stochastiques comme le Hill Climbing, le recuit accepte des solutions qui détériorent notre critère notamment lorsque la température est élevée. Cela lui permet ainsi d'échapper aux optima locaux.

La mise en œuvre de la méthode demande le réglage d'un certain nombre de paramètres à savoir :

- la **température initiale.**
- la **loi de décroissance de la température.**
- le **critère d'arrêt** (c'est-à-dire la valeur finale de T).
- le nombre de perturbations à chaque palier c'est-à-dire le critère de changement de température (**longueur de la chaîne de Markov** L_M).
- le **mécanisme de transformation** qui permet de passer d'une configuration à une autre.

[Laarhoven87] ont proposé différentes techniques de réglage de ces paramètres. Ces techniques assurent notamment la convergence de l'algorithme vers l'optimum global avec une probabilité tendant vers 1. Bien sûr ceci ne constitue qu'une preuve théorique et n'assure nullement en pratique que l'algorithme converge vers l'optimum en un temps fini.
Le réglage de ces paramètres s'appelle *Plan de Recuit* ou Cooling Schedule en anglais.

4.2.3 Plan de Recuit

Nous allons détailler différents plans de recuit mais pour cela il faut définir quelques grandeurs [Laarhoven87].

• Coût attendu de l'équilibre : cette grandeur peut être estimée comme étant la moyenne des coûts E de L_M différentes configurations $(x_i)_{i=1..L_M}$ testées de la chaîne de Markov à une température T.

$$\langle E(T)\rangle = \frac{1}{L_M}\sum_{i=1}^{i=L_M} E(x_i) = \overline{E}(T)$$

• Ecart carré attendu

$$\langle E^2(T)\rangle = \frac{1}{L_M}\sum_{i=1}^{i=L_M} E^2(x_i)$$

• Ecart type coût attendu

$$\sigma = \sqrt{\langle E^2(T)\rangle - \langle E(T)\rangle}$$

A Température initiale

Si la température initiale est trop basse, les nouvelles configurations ne seront acceptées que si elles améliorent le critère. Par conséquent, l'algorithme risque de converger vers un optimum local. La température initiale doit donc être suffisamment élevée pour permettre à toutes les configurations d'être acceptées et ainsi la région de l'optimum global pourra être localisée.
[Kirkpatrick83] a proposé une solution empirique : choisir une valeur de température initiale et générer un certain nombre de configurations. Ensuite, tant que le nombre de configurations acceptées est inférieur à un certain taux proche de 1, on multiplie par 2 la température et recommence le processus.

B Décroissance de la température

La loi de décroissance de T doit être choisie de telle manière que de petites chaînes de Markov assurent l'équilibre thermique. [Kirkpatrick83] et [Romeo85] ont proposé une loi linéaire où $T_{k+1}=\alpha T_k$ avec $\alpha \in [0.5 , 0.99]$.

[Huang86] base la loi de décroissance sur différentes moyennes de coûts entre deux chaînes de Markov consécutives, en utilisant l'équation :

$$\frac{\partial}{\partial lnT}\langle E(T)\rangle = \frac{\sigma^2(T)}{T}$$

et à l'aide d'une approximation de la dérivée par différences finies il obtient :

$$T_{k+1} = T_k exp\left(\frac{T_k\left(\langle E(T_{k+1})\rangle - \langle E(T_k)\rangle\right)}{\sigma^2(T_k)}\right)$$

Enfin, d'après [Huang86], pour maintenir l'équilibre il est nécessaire que le coût moyen entre deux chaînes de Markov consécutives soit inférieur à l'écart type du coût c'est-à-dire $\langle E(T_{k+1})\rangle - \langle E(T_k)\rangle = \lambda\sigma(T_k)$ avec $\lambda<1$. Ce qui donne une décroissance exponentielle de la température:

$$T_{k+1} = T_k exp\left(\frac{\lambda T_k}{\sigma(T_k)}\right)$$

D'autres schémas de refroidissement ont été proposés [Laarhoven87] [Lundy86] [Otten84]. Ils ont pour caractéristique commune de ne pas décroître plus rapidement que $\frac{T_0}{ln(k)}$ car ceci assure la convergence théorique - temps infini - vers l'optimum global [German84].

C Longueur des chaînes de Markov : L_M

Il faut, qu'à chaque palier de température T_k, la chaîne de Markov générée soit suffisamment longue pour que l'équilibre thermique soit atteint.

Un choix simple pour L_M consiste à arrêter la chaîne lorsqu'un certain nombre de configurations ont été acceptées. C'est-à-dire, lorsque le rapport nombre d'acceptations sur nombre de configurations refusées (**na/nr**) est supérieur à une certaine valeur. Cependant, lorsque T_k tend vers 0 la probabilité pour qu'une nouvelle configuration soit acceptée se réduit dramatiquement. En se basant sur le critère **na/nr** on obtiendrait alors des chaînes de Markov de longueur infinie. Pour éviter ce type de problème, on choisira une longueur maximale $\overline{L}$ pour L_M qui dépend elle, de façon polynomiale de la taille du problème.

D'une manière générale, la longueur L_M des chaînes de Markov est intimement liée au type de refroidissement utilisé pour mettre à jour le paramètre de contrôle T. Intuitivement, de larges

décréments de T_k nécessitent de plus grandes chaînes de Markov pour pouvoir atteindre l'équilibre thermique. L'équilibre thermique étant caractérisé par la convergence de la distribution des énergies, calculées sur les différentes configurations de la chaîne de Markov, vers une loi de distribution normale [Isaacson74][Seneta81]. En revanche, de faibles valeurs de décréments de T_k permettent l'utilisation de petites chaînes de Markov. Il y a donc clairement un compromis à faire pour d'une part converger vers l'optimum global et d'autre part éviter des temps de convergence trop importants.

D Création d'une nouvelle configuration dans la chaîne de Markov

Les principales méthodes de transformation ont été mises au point pour satisfaire les problèmes combinatoires [Kirkpatrick83]. Dans ces problèmes, une configuration voisine correspond à une permutation dans la liste des cas possibles.
Pour les problèmes continus, les transformations correspondent à de "petites" perturbations aléatoires sur les variables du problème. En pratique, si n est la dimension du problème, on construit un vecteur Δx de n variables aléatoires Δx et, à partir de la configuration x_i, on génère une nouvelle configuration par $x_{i+1} = x_i + \Delta x$ [Ingber92].
Pour simuler les variables aléatoires, on utilise une loi de distribution stochastique représentée par une densité g. Par exemple la distribution Gaussienne :

$$g(x) = (2\pi T_k)^{-n/2} exp\left(\frac{-\Delta x^2}{(2T_k)}\right)$$

où T_k est le palier de température courant.
Dans [Szu87], les auteurs ont utilisé une distribution de Cauchy définie par la densité :

$$g(x) = \frac{T_k}{\left(\Delta x^2 + T_k^2\right)^{(n+1)/2}}$$

Cette distribution est intéressante car elle permet l'utilisation d'une loi de décroissance de T du type $T_k = \frac{T_0}{(\lambda + \alpha k)}$ qui est plus rapide que la décroissance logarithmique nécessaire jusqu'à présent pour assurer la convergence vers l'optimum global.

E Critère d'arrêt

L'algorithme s'arrête lorsque plus d'amélioration sensible de la solution n'est obtenue ou lorsqu'une certaine valeur de la température est atteinte. On pourra par exemple utiliser un critère:
$\frac{\langle E \rangle - E_k}{\langle E \rangle} \leq \varepsilon$ où E_k est la valeur de la fonction à l'itération courante et ε un nombre << 1 qui dépend de la précision souhaitée.

5 VALIDATIONS

Le comportement des divers algorithmes d'optimisation a été étudié en utilisant une suite de fonctions tests. Cette suite regroupe, entre autres, des problèmes suggérés par les travaux de [DeJong75] sur les algorithmes génétiques. L'intérêt de ces fonctions est de présenter les difficultés classiques des problèmes d'optimisation de façon isolée. En comparant les divers algorithmes sur ces cas tests, les forces et les faiblesses de chacun d'entre eux pourront être estimées. Bien sûr, ces résultats sont obtenus pour des implantations bien particulières de ces algorithmes : d'autres implantations avec d'autres paramètres pourront donner des résultats différents.

5.1 Fonctions testées et leurs principales caractéristiques*

Sphère : Le *rêve* pour les algorithmes d'optimisation. Il s'agit d'un problème unimodal, convexe, continu et bien symétrique.

$$Min\, f_1(x) = \sum_{i=1}^{i=n} x_i^2 = f_1(0,\cdots,0) = 0$$

avec $-5.12 \le x_i \le 5.12$

Sphère

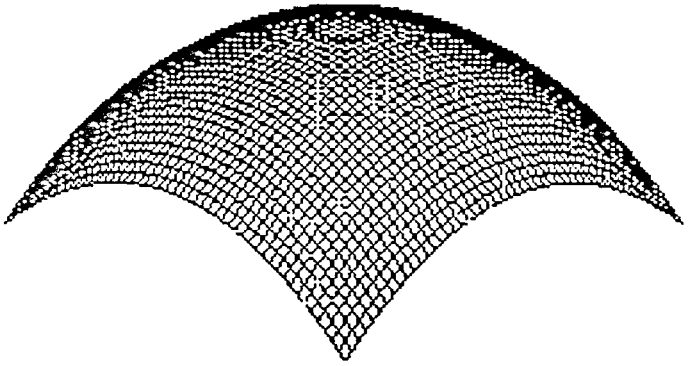


Figure I.7: Vue de la fonction f_1 pour n=2

Rosenbrock : Le *cauchemar* : le problème est aussi unimodal mais l'optimum se trouve dans une région très étroite à l'intersection de vallées peu pentues.

$$Min\, f_2(x) = \sum_{i=1}^{i=n} 100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2 = f_2(1,\ldots,1) = 0$$

avec $-5.12 \le x_i \le 5.12$

* Les surfaces présentées ont été modifiées par des rotations pour une plus grande clarté

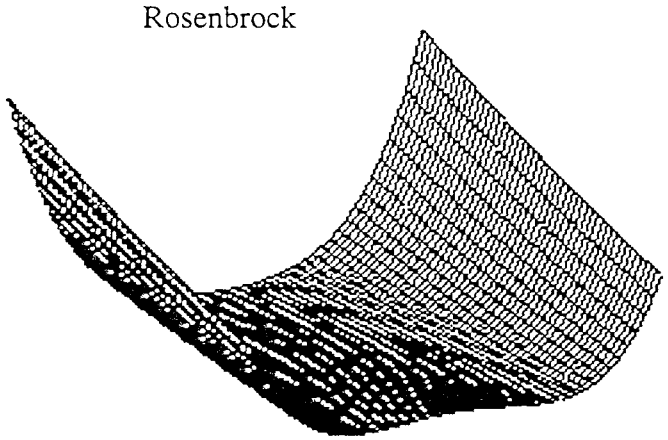


Figure I.8: Vue de f_2 pour n=2

Vague Sinusoïdale : cette fonction ne dépend que de deux paramètres mais possède un grand nombre d'optima locaux.

$$Max\, f_3(x) = 0.5 - \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2} = f_3(0,\ldots,0) = 1$$

avec $-100 \le x_i \le 100$

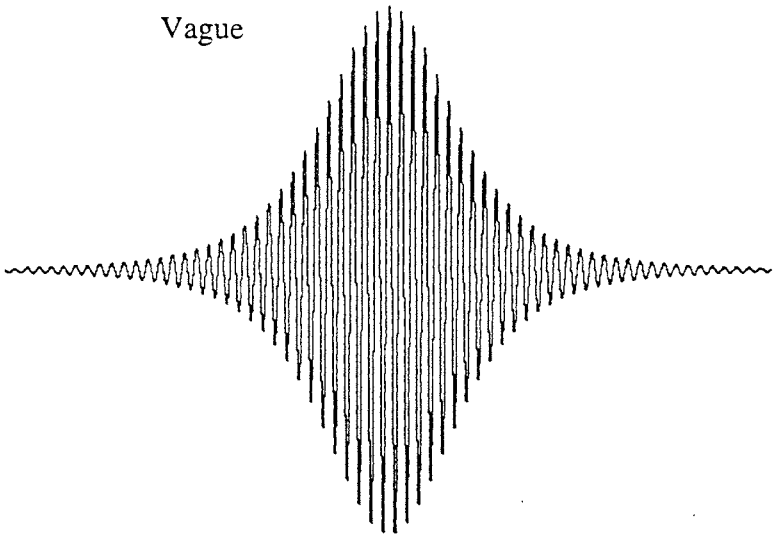


Figure I.9: Vue de la fonction f_3 sur une des dimensions

Rastrigin : Il s'agit d'une fonction possédant 10^n optima locaux. Elle permettra, comme la précédente, de tester la capacité des algorithmes stochastiques à localiser l'optimum global.

$$Min\, f_4(x) = 10n + \sum_{i=1}^{i=n}\left(x_i - 2.5\right)^2 - 10\cos\left(2\pi\left(x_i - 2.5\right)\right) = f_4(2.5,\ldots,2.5) = 0$$

avec $-5.12 \le x_i \le 5.12$

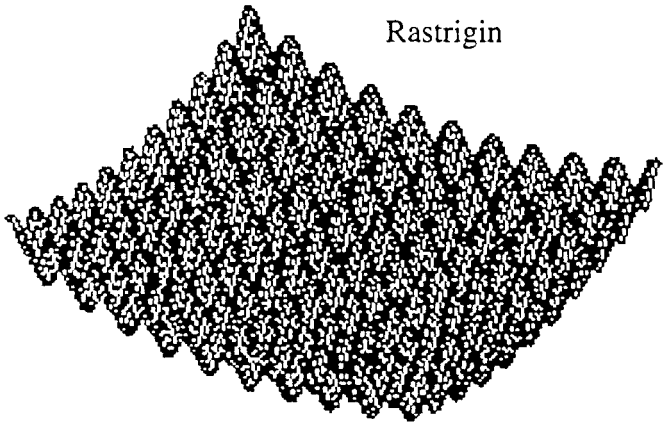


Figure I.10: Vue de la fonction f_4 pour n=2

Rosen Susuki : Fonction unimodale avec trois contraintes non linéaires.

$$Min\, f_5(x) = x_1^2 + x_2^2 + 2x_3^2 + x_4^2 - 5x_1 - 5x_2 - 21x_3 + 7x_4 + 100 = f_5(0,1,2,-1) = 56$$

$$\begin{cases} g_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_1 - x_2 + x_3 - x_4 - 8 \\ g_2(x) = x_1^2 + 2x_2^2 + x_3^2 + 2x_4^2 - x_1 - x_4 - 10 \\ g_3(x) = 2x_1^2 + x_2^2 + x_3^2 + 2x_1 - x_2 - x_4 - 5 \end{cases}$$

$$\text{avec} \begin{cases} 0 \le x_i \le 10 \quad pour \quad i = 1,2,3 \\ -1 \le x_4 \le 10 \end{cases}$$

Palier : Fonction discontinue.

$$Min\, f_6(x) = 6n + \sum_{i=1}^{i=n} entier(x_i) = f_6([-5.12,-5],\ldots,[-5.12,-5]) = 0$$

avec $-5.12 \le x_i \le 5.12$

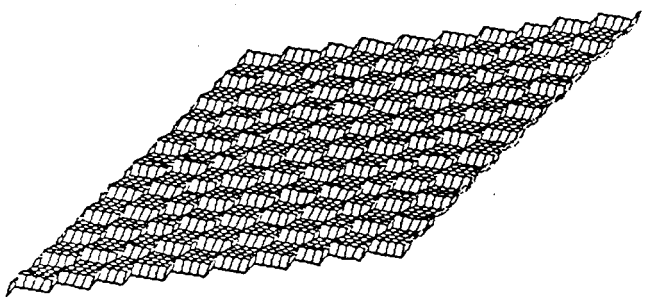


Figure I.11: Vue de la fonction f_6 pour n=2

5.2 Tests effectués

Les divers algorithmes étudiés dans la section précédente ont été comparés sur ces fonctions tests. L'algorithme de Recuit simulé a été "récupéré" dans le domaine public [SA]. Nous nous bornerons à le présenter dans le cadre de confrontations directes mais nous n'étudierons pas sa sensibilité vis à vis de ses paramètres.
Les critères retenus pour valider les méthodes d'optimisation sont :
- *Le nombre d'évaluations* de fonctions **neval** nécessaire pour atteindre l'optimum. Dans le cas de la méthode de Lagrangien Augmenté neval comptabilise à la fois les évaluations de l'objectif et les évaluations des gradients.
- *La qualité de l'optimum* c'est-à-dire la précision obtenue sur les valeurs des coordonnées du point optimal ainsi que la précision sur la valeur de la fonction à l'optimum.

Dans le cas des méthodes stochastiques il est impératif de procéder à un certain nombre d'exécutions de l'algorithme car, par exemple, une mutation pourrait trouver "par hasard" l'optimum. Nous avons donc relancé, pour chaque fonction test, le Recuit Simulé et l'Algorithme Génétique 30 fois et ceci constituera une mesure supplémentaire pour valider ces algorithmes. La valeur **nopt** désignera le nombre d'essais (sur 30) pour lesquels l'algorithme a effectivement localisé l'optimum avec une précision déterminée. Pour les algorithmes stochastiques, **neval** désignera le *nombre moyen* d'évaluations de l'objectif nécessaire pour atteindre l'optimum (sur **nopt** succès).
Le grand nombre de paramètres de l'algorithme génétique ne doit pas faire oublier qu'il nous faut déterminer des réglages qui ne soient pas trop disparates suivant la fonction traitée. En effet, lorsque nous aurons à traiter des problèmes réels nous n'aurons pas autant de liberté pour faire varier les valeurs de paramètres (à cause du coût des évaluations).

Notations et paramètres

- **MLA** : Méthode de Lagrangien Augmenté avec une méthode BFGS pour la résolution des problèmes sans contraintes.

- **AG** : Algorithme Génétique. L'algorithme utilise un codage de Gray de l'ensemble des variables, une sélection linéaire de *Baker*, un programme d'échantillonnage de *Brindle* et une mutation uniforme. Les paramètres de la méthode laissés variables sont:

nbpop	: Taille de la population.
nbgen	: Le nombre maximum de générations autorisées.
tsel	: Valeur MAX de la sélection linéaire.
txov	: Le type d'opérateur de croisement.
pc	: La probabilité de l'opérateur de croisement.
pm	: La probabilité de l'opérateur de mutation.

- **RS** : Recuit Simulé. La version implantée utilise un décrément de température en *T/k* avec des perturbations stochastiques de type *Cauchy*. La valeur initiale de la température est déterminée de façon à ce que toutes les transitions, même celles dégradant l'objectif soient acceptées. Le nombre de transitions testées à chaque palier de température peut être réglé par l'utilisateur (par défaut dans [SA], il vaut 20). La longueur des chaînes de Markov peut être aussi modifiée: un certain nombre (20 par défaut) de perturbations sont effectuées et lorsque un nombre **nr** de refus consécutifs a été atteint, on change de palier de température. Précisons que comme nous n'avons pas mis au point ce "code", peu de modifications sur les valeurs par défaut des paramètres ont été effectuées. Les paramètres gardés par défaut sont:

a : Paramètre de la décroissance de température $T_{k+1} = \frac{T_k}{1+ka}$ a=0.1

k_B : Facteur de Boltzmann pour l'acceptation des nouvelles transitions de la chaîne de Markov $k_B = 1$.

Les paramètres laissés variables sont:

k_{max} : Nombre maximum de paliers de température.

it_pal : Nombre de perturbations maximum par palier de température.

nr : Nombre de refus consécutifs indiquant qu'il faut changer de palier.

5.3 Algorithme génétique

La table I.3 présente les performances de l'algorithme génétique sur l'ensemble des fonctions tests. Au valeurs **nopt** et **neval**, nous avons ajouté la dimension de l'espace continu des solutions **n** et le nombre de bits l codant un individu et donnant la dimension de l'espace discrétisé. Dans tous les cas, la précision souhaitée sur les variables à l'optimum était de 3 chiffres significatifs après la virgule sauf pour la fonction f_5 où on s'est contenté d'une erreur relative sur la valeur de la fonction à l'optimum de 10^{-2}. Nous avons aussi rajouté comme critère d'arrêt un nombre maximum d'évaluations fixé arbitrairement à 25 000.

Fonction	n	dim. 2^l	nopt	neval
f_1	3	42	30	1809
f_2	2	28	30	8705
f_3	2	36	30	8910
f_4	10	140	0	-
f_5	4	56	30	7136
f_6	5	70	30	1800

Table I.3: Performances de l'algorithme génétique sur les 6 fonctions tests

Ce tableau démontre la faculté qu'on les algorithmes génétiques à localiser l'optimum sur des problèmes de nature très variée. Il est notamment intéressant de noter que ces résultats ont été obtenus pour des valeurs de paramètres de l'algorithme sensiblement identiques pour les différentes fonctions

(Table I.4). La probabilité de mutation prend des valeurs de l'ordre de **1/*l*** et la probabilité de croisement est assez élevée dans tous les cas. Le réglage de la *pression de sélection* est assuré ici à l'aide du paramètre **tsel** qui prend des valeurs dans l'intervalle [1.6, 1.8]. Enfin, pour tous les tests l'opérateur de croisement était le crossover 2-points.

Fonction	pc	pm	tsel	nbpop	txov
f_1	*0.7*	*0.01*	*1.7*	*26*	*2-points*
f_2	*0.7*	*0.02*	*1.7*	*50*	*2-points*
f_3	*0.9*	*0.02*	*1.6*	*50*	*2-points*
f_4	*0.7*	*0.01*	*1.7*	*70*	*2-points*
f_5	*0.7*	*0.018*	*1.7*	*30*	*2-points*
f_6	*0.8*	*0.01*	*1.8*	*30*	*2-points*

Table I.4: Valeurs des paramètres de l'AG pour les tests de la Table I.3

La fonction de Rastrigin s'est révélée très difficile pour notre algorithme génétique qui n'a jamais pu localiser l'optimum ni même sa région (précision plus faible mais le deuxième optimum est passé !). Pour les valeurs des paramètres données dans la table I.4, la valeur moyenne (sur les 30 exécutions) obtenue pour l'objectif au bout des 25 000 évaluations était de 5.51 c'est-à-dire sur des optima pas trop éloignés.

L'autre fonction multimodale a donné des résultats bien plus concluants. Notamment pour cette fonction, le deuxième optimum est 0.990 et pour les valeurs des paramètres données dans la table I.4 l'algorithme a franchi ce pic en 2845 évaluations en moyenne. Bien sûr f_3 ne possède pas autant d'optima que la fonction de Rastrigin pour n=10, cependant nous avons testé cette dernière pour n=5 et n=3 c'est-à-dire des fonctions possédant respectivement 100 000 et 1000 optima. Les résultats sont présentés dans la table I.5 dans laquelle nous avons aussi fait figurer le nombre moyen d'évaluations pour atteindre la région de l'optimum global (**Région**).

Fonction	n	dim.	pc	pm	tsel	nbpop	txov	nopt	Région	neval
f_4	*5*	*70*	*0.9*	*0.014*	*1.6*	*100*	*2-points*	*27*	*8121*	*14573*
f_4	*5*	*70*	*0.9*	*0.014*	*1.6*	*70*	*2-points*	*22*	*9182*	*12765*
f_4	*5*	*70*	*0.9*	*0.014*	*1.6*	*50*	*2-points*	*21*	*10965*	*13830*
f_4	*3*	*42*	*0.9*	*0.024*	*1.6*	*60*	*2-points*	*30*	*3558*	*6863*

Table I.5: Performances de l'AG sur fonction de Rastrigin pour différentes dimensions

On constate que, dans ces cas, l'algorithme génétique localise l'optimum notamment pour n=3. Pour n=5 les échecs sont dus à une convergence vers le deuxième optimum pour lequel la fonction vaut 1. Là encore le nombre d'évaluations nécessaire pour localiser la région de l'optimum global est bien plus faible et on peut remarquer que la localisation est plus rapide pour une large population qui implique beaucoup plus de schémas dans la recherche. On peut d'ores et déjà conclure que pour l'algorithme génétique le coût (en nombre d'évaluations) pour obtenir une bonne précision est relativement

important. En revanche la localisation de la région de l'optimum ne nécessite pas autant d'efforts de calcul.

En ce qui concerne les fonctions unimodales, l'algorithme génétique n'a pas eu de problèmes pour localiser l'optimum mais là encore, le nombre d'évaluations pour obtenir la précision que nous souhaitions est assez élevé surtout si on le compare à celui obtenu avec une méthode déterministe (voir par ailleurs).

Nous avons aussi regardé le comportement de l'algorithme sur la fonction f_3 pour un opérateur de croisement uniforme au lieu de l'opérateur 2-points (Table I.6) tout en conservant les autres paramètres identiques. Ces essais ont été réalisés sur deux tailles de population: 30 et 50.

Fonction	pc	pm	tsel	nbpop	txov	nopt	neval
f_3	*0.9*	*0.03*	*1.6*	*50*	*2-points*	*30*	*8910*
f_3	*0.9*	*0.03*	*1.6*	*50*	*Uniforme*	*27*	*7380*
f_3	*0.9*	*0.03*	*1.6*	*30*	*2-points*	*25*	*7350*
f_3	*0.9*	*0.03*	*1.6*	*30*	*Uniforme*	*26*	*6395*

Table I.6: Comparaison des performances du croisement uniforme et du croisement 2-points

Les résultats sont sensiblement équivalents pour les deux opérateurs. On peut remarquer cependant que pour une population plus faible (30) et une valeur **nopt** identique le nombre d'évaluations moyen nécessaire pour atteindre l'optimum est inférieur dans le cas de l'opérateur de croisement uniforme. Ceci confirme ce que nous avions dit précédemment au sujet de cet opérateur: il permet bien une exploration de l'espace des solutions plus importante dans le cas des populations de faible taille.

Nous avons jusqu'à présent conservé dans tous les tests des probabilités de croisement et de mutation identiques. Nous avons donc fait varier les probabilités **pc** et **pm** pour regarder l'influence de ces paramètres sur la recherche de l'optimum. Les tests ont été réalisé sur les fonctions f_2 et f_3. La table I.7 comptabilise les succès (**nopt**) pour les différentes valeurs de **pc** et les courbes de la figure I.12 montrent l'évolution de f_3 pour le meilleur individu en fonction du nombre d'évaluations. On constate que si l'algorithme localise bien l'optimum global pour les différentes valeurs de **pc**, sa convergence est grandement améliorée pour des valeurs de **pc** élevées. En effet, pour **pc** = 0.2, la région de l'optimum global ($f_3 > 0.990$) n'est découverte qu'après 10 000 évaluations alors que pour **pc** = 0.8 cette région est atteinte rapidement.

pc	nopt
0.8	*30*
0.6	*30*
0.4	*30*
0.2	*26*

*Table I.7: Valeurs de **nopt** pour différentes valeurs de pc sur la fonction f_3 (**pm**=0.03 , **tsel**=1.6, **nbpop**=50, croisement **2-points**)*

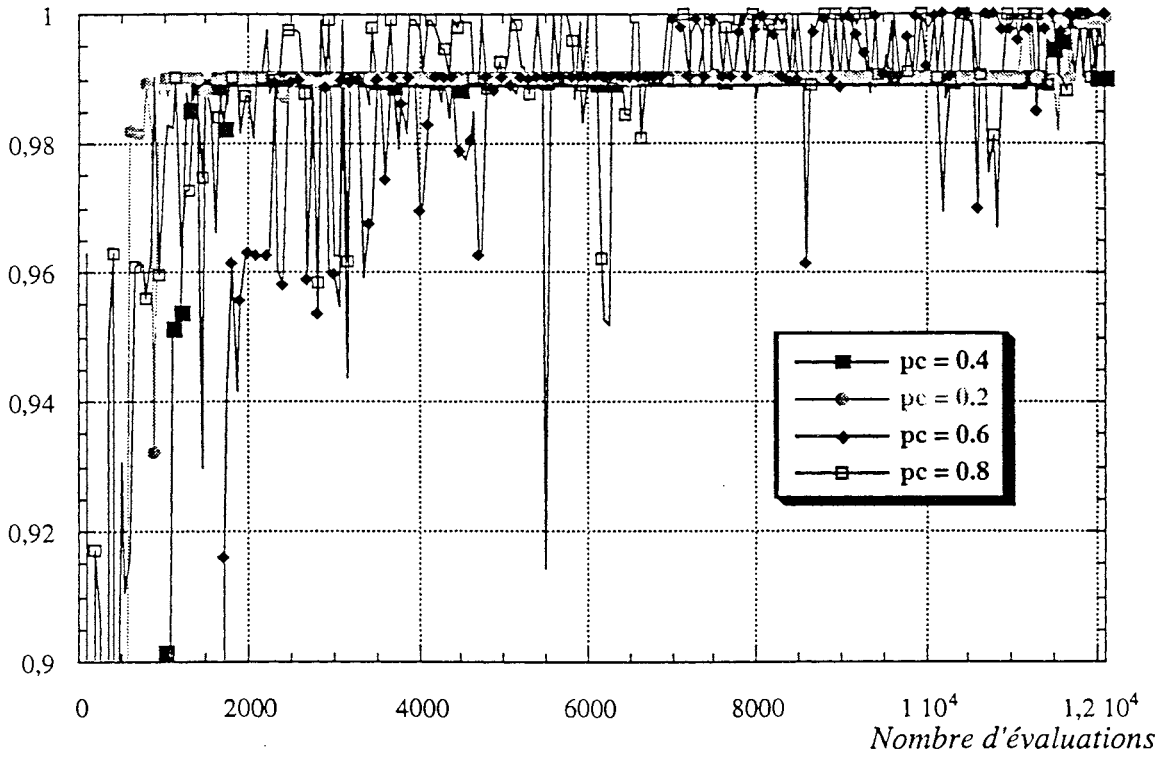


Figure I.12: Evolution de la fonction f_3 pour différentes valeurs de **pc**

La table I.8 présente les résultats obtenus pour différentes valeurs de la probabilité de mutation sur la fonction f_2. Sur la figure I.13 nous avons tracé les valeurs de f_2 pour le meilleur individu en fonction du nombre d'évaluations. On constate que l'opérateur de mutation est absolument nécessaire pour approcher avec une bonne précision l'optimum. En revanche si **pm** est trop élevé alors les bons schémas sont rapidement détruits et la recherche ne peut être efficace. Il semble donc que des valeurs proches ou légèrement inférieures à *1/l* donnent les meilleurs résultats.

pm	**nopt**
0.002	*3*
0.008	*15*
0.015	*27*
0.03	*30*
0.1	*0*

Table I.8: Valeurs de **nopt** *pour différentes valeurs de* **pm** *sur la fonction f_2 (**pc**=0.7 , **tsel**=1.6, **nbpop**=50, croisement* **2-points**)

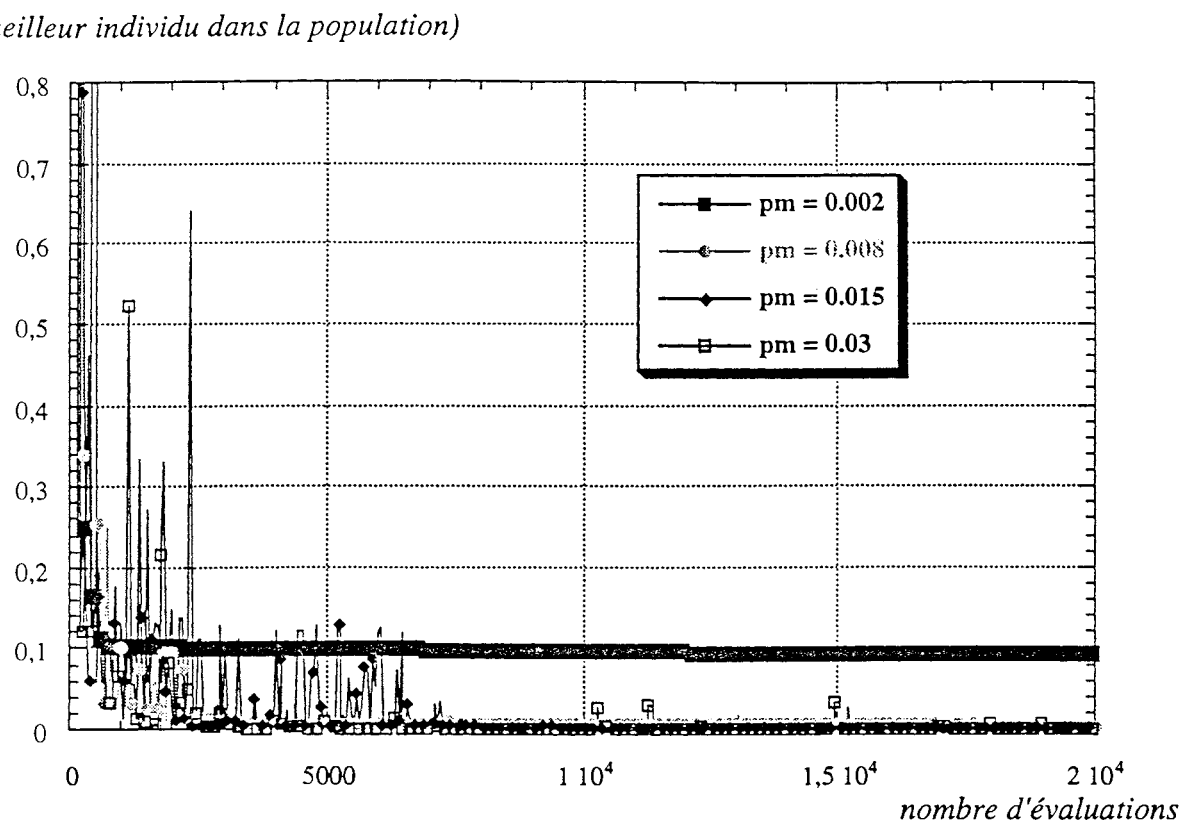


Figure I.13: Evolution de la fonction f_2 pour différentes valeurs de **pm**

5.4 Recuit Simulé

La table I.9 résume les résultats que nous avons obtenus avec l'algorithme de Recuit Simulé. Cet algorithme donne sensiblement les mêmes résultats que l'algorithme génétique sur les fonctions unimodales. Sur les fonctions multimodales, l'algorithme de Recuit n'a pu trouver l'optimum avec une précision de 10^{-3} sur les variables que pour 17 essais sur 30 et pour les autres il est resté piégé sur l'optimum local 0.99. Cependant, il faut prendre ces résultats avec prudence car il s'agit uniquement d'une version "domaine public" sur laquelle nous n'avons effectué que peu de changements et elle n'est présentée ici que pour servir d'élément de référence à notre algorithme génétique. D'ailleurs sur la fonction de Rastrigin pour n=5, l'optimum global est localisé avec la précision désirée dans 26 essais sur 30 et avec 21665 évaluations en moyenne. Dans ces tests nous avions pris k_{max}=350, **it_pal**=100 et **nr**=50 et pour la fonction de Rosen Susiki nous avons traité les contraintes à l'aide d'une fonction de pénalisation extérieure.

Fonction	n	nopt	neval
f_1	3	30	2508
f_2	2	30	11001
f_3	2	17	18720
f_4	10	2	29268
f_5	4	30	5827
f_6	5	30	1636

Table I.9: Performances du recuit simulé sur les 6 fonctions tests

5.5 Lagrangien Augmenté

Les tables I.10, I.11, I.12 présentent les performances de l'algorithme de Lagrangien Augmenté sur les 3 fonctions unimodales et dérivables. Le nombre d'évaluations prend en compte le gradient et dans ces tests nous avons fait figurer la solution finale. On s'aperçoit que le nombre d'évaluations est bien moins important que dans le cas des méthodes stochastiques et que la solution finale est bien plus précise. D'ailleurs, compte tenu du caractère unimodal de ces fonctions, l'optimum est trouvé quel que soit le point de départ, cependant, suivant la valeur de celui ci le nombre de calculs peut être plus ou moins important.

Fonction	n	Point initial	Point final	neval
f_1	3	(5, -5, 5)	(10^{-14}, 10^{-14}, 10^{-14})	6+18
f_1	3	(1, 5, -5)	(10^{-7}, 10^{-8}, 10^{-8})	6+18
f_1	3	(2, 3, -4)	(10^{-7}, 10^{-7}, 10^{-7})	5+15

Table I.10: Performance du Lagrangien augmenté sur la fonction f_1

Fonction	n	Point initial	Point final	neval
f_2	2	(2, 3)	(0.999999, 0.999999)	38+76
f_2	2	(5, -5)	(1.000000, 0.999999)	83+166
f_2	2	(5, 0.01)	(1.0000000,0.9999999)	74+148

Table I.11: Performance du Lagrangien augmenté sur la fonction f_2

Fonction	n	Point initial	Point final	neval
f_5	4	(5, 5, 5, 5)	(10^{-6}, 0.99998, 2.00000, -0.99998)	54+216
f_5	4	(2, 2, 2, 2)	(10^{-6}, 1.000000, 1.99999, -0.99999)	46+184
f_5	4	(5, 0, 6, 9)	(10^{-5}, 0.99998, 1.99998, -1.00003)	57+228

Table I.12: Performance du Lagrangien augmenté sur la fonction f_5

6 CONCLUSION

Dans ce chapitre, nous avons présenté une méthode de Lagrangien Augmenté pour l'optimisation sous contraintes non linéaires. Cette méthode possède la particularité de ne pas être sensible aux problèmes de conditionnement habituellement rencontrés dans les méthodes de pénalités. Sur les fonctions testées qui étaient unimodales et dérivables, elle a trouvé l'optimum avec un nombre de calculs de fonctions bien plus faible que pour les méthodes stochastiques.
En revanche, dans le cas des fonctions présentant plusieurs optima la méthode de Lagrangien Augmenté n'est pas en mesure de localiser l'optimum global, à moins d'un heureux concours de circonstances (point initial proche). C'est pour cette classe de fonctions que nous avons introduit les algorithmes stochastiques.

L'algorithme génétique se base sur les principes dérivés de l'évolution des espèces et utilise un codage de l'espace des paramètres ainsi que des opérateurs très simples comme le croisement et la mutation. Cet algorithme possède le double avantage de ne pas nécessiter le calcul du gradient et surtout d'être capable de localiser l'optimum global. Cette dernière caractéristique a été confirmée par les résultats obtenus sur les fonctions multimodales testées. Les tests ont été de plus réalisés avec des valeurs de paramètres de l'algorithme presque identiques quelle que soit la nature de la fonction test. Ceci est important puisque pour des problèmes "réels" nous n'aurons pas la chance de pouvoir procéder à de nombreux essais à cause du coup des évaluations.

L'algorithme de Recuit Simulé a été développé par analogie entre le procédé de recuit en métallurgie des solides et l'optimisation mathématique. Les performances obtenues par cet algorithme sur les différentes fonctions tests ont été légèrement inférieures à celle de l'algorithme génétique mais il ne s'agissait ici que d'une version du domaine public sur laquelle nous n'avons pas effectué autant d'essais. Notre but était de comparer l'algorithme génétique à une méthode d'optimisation stochastique ayant déjà fait ses preuves en Electromagnétisme.
Quoiqu'il en soit, les tests réalisés sur ces algorithmes nous permettent de tirer quelques enseignements sur leurs comportements:

- Les algorithmes de Recuit Simulé et Génétiques sont capables de localiser l'optimum global.
- Sur les fonctions unimodales, l'algorithme de Lagrangien Augmenté (ou BFGS dans le cas sans contraintes) converge vers l'optimum en nécessitant bien moins de calculs que les méthodes stochastiques pour une même précision.
- Sur les problèmes multimodaux, l'algorithme génétique a nécessité de nombreux calculs supplémentaires pour obtenir la précision souhaitée alors qu'il avait déjà localisé la région de l'optimum global.

Il faut ajouter que sur la fonction de Rastrigin pour n=10 aucun des deux algorithmes stochastiques n'a été en mesure de localiser la région de l'optimum global.

Tout ceci nous amène à nous demander:

- si il est possible de diminuer encore le nombre d'évaluations pour atteindre la région de l'optimum.
- si il est possible d'améliorer la précision de la solution obtenue.
- si il est possible, moyennant un coût raisonnable en évaluations, de trouver l'optimum global pour des problèmes comme celui de Rastrigin en dimension 10.

C'est à ces questions que nous avons voulu répondre dans le prochain chapitre pour pouvoir ensuite nous attaquer à des problèmes réels avec un algorithme performant.

Chapitre II

Vers un Algorithme performant...

Chapitre II

Vers un Algorithme performant...

1 INTRODUCTION

Dans le chapitre précédent nous avons pu comparer les forces et les faiblesses des divers algorithmes en étudiant leur comportement sur certaines classes de fonctions. Cependant, il reste quelques questions en suspens et notamment nous aimerions encore améliorer les performances de l'algorithme génétique en diminuant le nombre d'évaluations pour atteindre l'optimum et en augmentant la précision de la solution obtenue.

Dans ce chapitre nous étudierons la possibilité d'un couplage entre un algorithme génétique et un algorithme déterministe. Ce type d'**hybridation**, lorsqu'elle est possible, nous permettra d'utiliser au mieux les avantages de ces deux méthodes, à savoir :

- La capacité d'un algorithme génétique à localiser la région de l'optimum global.
- La rapidité de convergence, la précision et le faible coût en termes d'évaluations que possède un algorithme déterministe lorsque le point de départ est proche de la solution optimale (globale !).

Se pose alors la question fondamentale de savoir à quel instant il est opportun de commute de l'algorithme génétique vers l'algorithme déterministe. Nous étudierons le critère d'une telle commutation.

Avant cela nous débuterons par quelques considérations sur l'algorithme génétique et notamment sur la viabilité du codage binaire. En effet, même si celui ci semble bien se prêter aux problèmes pour lesquels les structures manipulées sont des variables réelles continues, il peut se révéler inutilisable pour des problèmes bien plus complexes. Cette question du codage nous renvoie à des considérations évoquées dans le chapitre précédent : la recherche est-elle bien le fait de la concaténation d'informations élémentaires ou le fruit d'un équilibre entre exploitation et exploration? Nous exposerons donc quelques principes pour implanter un algorithme génétique lorsque les structures manipulées ne permettent pas l'utilisation d'un codage binaire. Cette démarche nous conduira alors naturellement à un algorithme spécialisé pour l'optimisation de maillages CAO (Chapitre IV) mais elle pourra servir dans d'autres domaines de l'électrotechnique comme l'optimisation de réseaux par exemple. Nous proposerons aussi un algorithme génétique basé sur un codage réel qui sera utilisé lorsque les variables sont réelles et en électromagnétisme nous l'emploierons dans le cas où le critère est donné sous forme analytique (Chapitre III) ou pour l'optimisation de forme (Chapitre V). Bien sûr

pour ces types de problèmes, l'algorithme génétique classique pouvait convenir, mais l'approche **codage réel** a donné des performances supérieures à celle de l'approche binaire sur les fonctions tests du chapitre précédent.

Enfin pour terminer, nous évoquerons les possibilités de **parallélisation** de l'algorithme génétique. En effet une parallélisation peut se révéler très intéressante lorsque l'évaluation du critère devient très coûteuse. Le portage du code séquentiel en code parallèle peut d'ailleurs être grandement simplifié par l'utilisation d'un langage de programmation par transfert de messages de type PVM (Parallel Virtual Machine) qui permet à l'aide de fonctions très simples d'emploi, de programmer de façon transparente dans un environnement hétérogène de machines.

2 VIABILITE DU CODAGE BINAIRE

Les algorithmes génétiques classiques sont considérés comme des algorithmes robustes c'est-à-dire qu'ils peuvent s'appliquer à une large gamme de problèmes, sans nécessiter d'informations sur le problème traité autres que l'évaluation du critère. Cependant, cette absence d'informations supplémentaires quant à la nature du problème peut aussi constituer une faiblesse de l'algorithme génétique, notamment dans des cas où les contraintes sont non "triviales". Pour illustrer ceci, on peut penser au célèbre problème du voyageur de commerce. Dans ce problème, un voyageur doit parcourir toutes les villes situées sur son territoire. Il doit passer dans chacune d'entre elles une unique fois et revenir, en fin de parcours, à la ville de départ. On se propose alors de répondre à la question : *moyennant un coût de voyage donné entre chaque ville, quel est le parcours optimal du voyageur de commerce pour minimiser le coût global ?* Dans un algorithme génétique classique, un individu représentera l'ensemble des villes parcourues par le voyageur de commerce. En utilisant les opérateurs de croisement et de mutation, on court le risque de passer la plupart du temps à évaluer des individus ne satisfaisant pas la contrainte c'est à dire des individus codant au moins 2 fois la même ville. De nombreux auteurs [Grefenstette85][Goldberg85][Davis85] ont donc proposé des algorithmes génétiques modifiés. Dans ces algorithmes le codage binaire est abandonné et les opérateurs de croisement et mutation sont spécialement redéfinis pour permettre de prendre en compte la contrainte du voyageur de commerce. En électronique, on peut être confronté à ce type de problème. Par exemple dans la recherche d'une disposition optimale des composants sur une carte VLSI.
Ceci ne constitue qu'une illustration mais ce type de problème, lié au codage binaire, se rencontre dans de nombreux domaines [Forrest85b][Coombs87][Glover87][Fox91][Vignaux91]. Ainsi, il vaudra mieux conserver un codage *naturel* des structures manipulées par notre problème et adapter en conséquence les opérateurs de croisement et de mutation. Les opérateurs génétiques devront alors satisfaire aux principes fondamentaux suivants :

- Le ***croisement*** est un opérateur qui ***échange de l'information*** entre 2 individus (ou plus !)
- La ***mutation*** est opérateur qui ***perturbe l'information*** contenue dans un individu.

En abandonnant le codage binaire on s'éloigne de la théorie de Holland mais on permet l'introduction d'informations supplémentaires dépendantes du problème qu'on se propose de traiter. Ces modifications donnent naissance à une nouvelle classe d'algorithmes génétiques que [Michalewicz94] nomme Evolution Program's (**EP's**) (Programmes d'évolution).
La figure II.1 illustre la différence entre l'approche classique des algorithmes génétiques et l'approche EP.
Nous utiliserons, dans le chapitre IV, une démarche EP pour la mise en œuvre d'un algorithme génétique spécialisé pour l'optimisation de maillages. Cette démarche devra être employée dès lors que les structures manipulées sont complexes.

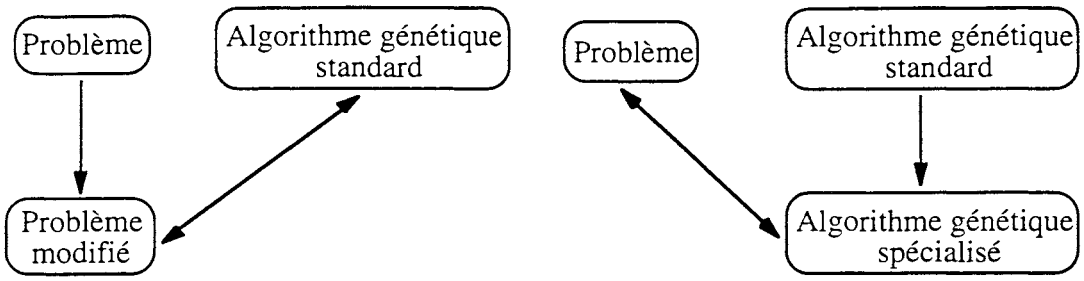


Figure II.1: Comparaison entre l'approche génétique ***Standard*** *et l'approche* ***Evolution Program***

En ce qui concerne l'optimisation de structures en électrotechnique, les algorithmes génétiques standards ont déjà permis de résoudre un certain nombre de problèmes, mais là encore une approche EP est envisageable. L'ensemble des paramètres de notre problème ne sera plus représenté par une chaîne binaire mais par un vecteur de nombres réels. Ce codage naturel des paramètres réels - codage réel -, est encore très discuté dans la communauté *"Génétique"* puisqu'il remet en cause la théorie des schémas de Holland. Dans la section suivante, nous présenterons un algorithme basé sur ce type de codage et il sera comparé à l'algorithme standard sur un ensemble de fonctions tests. Dans la suite cet algorithme sera utilisé pour traiter les problèmes d'optimisation de structures en électromagnétisme (Chapitre III et Chapitre V).

3 UN ALGORITHME GÉNÉTIQUE AVEC CODAGE RÉEL

3.1 Présentation de l'algorithme mis au point

3.1.1 Codage des paramètres

L'ensemble des variables $(x_i)_{i=1,n}$ est représenté par un vecteur : $\bar{x}=<x_1, x_2, \cdots, x_n>$ où chaque x_i est un nombre réel. Initialement chaque x_i est généré aléatoirement dans $\left[x_i^{min}, x_i^{max}\right]$. Les opérateurs de croisement et de mutation seront alors soigneusement définis pour conserver chacune des variables dans son domaine. La plupart de ces opérateurs s'inspirent d'ailleurs des opérateurs du codage binaire.

3.1.2 Opérateurs de Croisement

Des opérateurs simples du type 1-point, 2-points ou uniforme peuvent être implantés de la même manière que dans le cas d'un codage binaire. La seule différence réside dans le point de coupe qui doit être choisi entre deux variables du vecteur ce qui revient à permuter des variables entre 2 chaînes. La figure II.2 présente l'opérateur de croisement 1-point.

$$\begin{array}{l} \bar{x}=<x_1, x_2, \cdots, x_k \mid x_{k+1}, \cdots, x_n> \\ \bar{y}=<y_1, y_2, \cdots, y_k \mid y_{k+1}, \cdots, y_n> \end{array} \Longrightarrow \begin{array}{l} \bar{x}'=<x_1, x_2, \cdots, x_k, y_{k+1}, \cdots, y_n> \\ \bar{y}'=<y_1, y_2, \cdots, y_k, x_{k+1}, \cdots, x_n> \end{array}$$

Figure II.2: Croisement 1-point pour le codage réel

L'opérateur de croisement arithmétique (Fig. II.3) [Janikow91] est légèrement différent et a été implanté spécialement pour le codage réel. L'échange d'informations se fait au travers d'une moyenne arithmétique entre les 2 chaînes parents. Cet opérateur utilise un nombre aléatoire $a \in [0,1]$ qui garantit que les descendants auront bien leurs gènes x_i dans $\left[x_i^{min}, x_i^{max}\right]$.

$$\vec{x}' = a\vec{x} + (1-a)\vec{y}$$
$$\vec{y}' = (1-a)\vec{x} + a\vec{y}$$

Figure II.3: Croisement arithmétique

Cet opérateur peut d'ailleurs s'appliquer sur k parents. Pour cela il suffit de générer k nombres α_i tels que $\sum \alpha_i = 1$ et de générer les descendants de la manière suivante : $\vec{x}' = \sum \alpha_i \vec{x}_i$.

3.1.3 Opérateurs de mutation

Là encore, les opérateurs sont légèrement différents de ceux vus dans le chapitre précédent. Tant dans la manière de muter le gène que dans le choix de ce gène: l'information perturbée n'est plus un bit mais une variable de notre problème. On rencontre dans la littérature l'opérateur de mutation **uniforme**, l'opérateur de mutation **non uniforme** [Davis89] et un opérateur de mutation **aux bornes** [Janikow91].

La mutation uniforme sélectionne un gène k dans la chaîne $\vec{x} = < x_1, x_2, \cdots, x_k, \cdots, x_n >$ et génère la nouvelle chaîne $\vec{x}' = < x_1, x_2, \cdots, x_k', \cdots, x_n >$ où x_k' est une valeur aléatoire prise dans l'intervalle $\left[x_k^{min}, x_k^{max}\right]$. La mutation non uniforme génère aussi une nouvelle valeur sur le gène k sélectionné mais dans le cas d'un codage réel x_k' est donné par la formule :

$$x_k' = \begin{cases} x_k + \Delta\left(t, x_k^{max} - x_k\right) \\ \quad ou \\ x_k - \Delta\left(t, x_k - x_k^{min}\right) \end{cases}$$

où t représente la génération courante et $\Delta(t, y)$ une fonction qui retourne un nombre aléatoire dans $[0, y]$ et dont la probabilité de renvoyer un nombre proche de 0 augmente lorsque t augmente. Cette propriété de Δ permet à l'opérateur de chercher de façon uniforme dans l'espace de solutions dans les premières générations puis de façon plus "locale" en fin d'exécution. [Michalewicz94] a proposé pour Δ la fonction :

$$\Delta(t, y) = y\left(1 - r^{(1-t/NBGEN)^b}\right)$$

avec *NBGEN* le nombre maximum de générations autorisées, r un nombre aléatoire dans l'intervalle $[0, 1]$ et b un paramètre de l'opérateur qui définit le degré de non uniformité. Pour Δ on peut aussi prendre la fonction :

$$\Delta(t, y) = y \cdot r\left(1 - \frac{t}{NBGEN}\right)^b$$

Celle ci donne, d'après Michlewicz, de meilleurs résultats [Michalewicz94]
L'opérateur de mutation aux bornes sélectionne une des variables de la chaîne et fixe sa valeur à une des bornes de son domaine de variations. L'intérêt d'un tel opérateur est qu'il permet de tester des valeurs du domaine qui n'auraient pu être accédées par le simple fait des croisements présentés plus haut.

Jusqu'à présent les opérateurs Croisement - Mutation étaient liés : la mutation agissait sur toutes les chaînes issues du croisement des individus sélectionnés. [Davis89] a introduit le concept d'opérateurs indépendants. Une ou deux chaînes sont sélectionnées, puis soit l'opérateur de croisement, soit l'opérateur de mutation est appliqué et non les deux successivement comme c'était le cas précédemment. Dans notre algorithme génétique, nous avons implanté trois opérateurs de mutation (**uniforme, non uniforme, aux bornes**) et quatre opérateurs de croisement (**1-point, 2-points, Uniforme, Arithmétique**). Des poids leur ont été affectés et un opérateur est ensuite sélectionné à l'aide d'une Roue de Loterie biaisée. Il existe différentes façons de calculer ces poids. On peut par exemple les maintenir constants tout au long de l'exécution ou bien les adapter linéairement. Mais il faut garder à l'esprit qu'un opérateur de croisement est bien plus intéressant en début d'exécution lorsque l'exploration est prépondérante. Par contre, lorsque la population a convergé, seuls les opérateurs de mutation sont capables d'apporter un peu de diversité et de maintenir un semblant d'exploration. Ce genre de considérations empiriques est en général utilisé pour régler les poids des opérateurs, cependant [Davis89] a proposé une approche plus rationnelle : l'adaptation des poids se fait en fonction de la performance des individus générés par l'opérateur. Ainsi si un opérateur "travaille bien", il sera utilisé de manière plus fréquente. Malheureusement, il semble qu'il soit nécessaire de considérer tous les parents qui ont permis d'aboutir au dernier descendant. Ceci demande la mise en œuvre de structures assez lourdes et pour notre algorithme nous avons donc préféré nous en tenir à des considérations empiriques.

En plus des opérateurs vus plus haut, nous avons implanté un opérateur de recherche locale basé sur des perturbations de type gaussienne. Cet opérateur peut être vu comme un petit Hill-Climbing intégré dans l'algorithme génétique. Il agit sur un individu $\vec{x}$ et le perturbe suivant une loi normale $N(\vec{x}, \sigma\, Id)$ où Id représente le vecteur Identité et σ un nombre réel strictement positif. Brièvement, on peut décrire cet opérateur - pour une maximisation - comme présenté sur la figure II.4. Pour simuler le vecteur aléatoire gaussien $\vec{y}$ à l'aide de la loi $N(\vec{x}, \sigma\, Id)$ on utilisera, à partir de variables aléatoires uniformes, des formules en *log-sinus* [Bouleau86].
Cet opérateur rentre dans la classe des mutations puisqu'il ne perturbe qu'une seule chaîne. Mais il agit comme un algorithme de recherche locale à part entière : il tente en effet d'améliorer le point courant en effectuant un certain nombre de tirages. Si le nombre de tirages maximum est égal à 1 alors cet opérateur est équivalent à une mutation classique mais de loi gaussienne plutôt qu'uniforme. Pourquoi utiliser un tel opérateur ? En fait il permet à notre algorithme génétique de gravir le pic, s'il existe, de la région occupée par le point sélectionné. Intuitivement, cet opérateur apporte de l'information quant à la

topologie du problème car il est capable de gravir des optima dans certaines régions de l'espace de recherche. Bien entendu, cet opérateur ne devra pas être employé avec un poids très élevé : on risquerait alors de faire converger les individus vers un optimum local. En revanche en fin d'exécution, s'il est employé avec un taux important, il permet à l'algorithme génétique de se comporter comme un algorithme de recherche locale.

En fait l'algorithme génétique ne doit plus être vu comme une méthode d'optimisation bien spécialisée mais au contraire c'est l'ensemble des méthodes d'optimisation présentées précédemment qui sont des cas particuliers d'algorithme génétique. Ainsi, le Hill-Climbing est un algorithme génétique où la population se réduit à un individu, où l'opérateur de sélection va en direction des meilleurs et où le seul opérateur est un opérateur de mutation gaussien. En remplaçant l'opérateur gaussien par le gradient on obtient une méthode déterministe. Même le Recuit est un algorithme génétique: l'opérateur de sélection est du type Boltzmann et l'opérateur de mutation est un de ceux présentés au chapitre précédent (Cauchy par exemple). A partir de là, on peut se concentrer sur les opérateurs de la recherche et c'est donc tout naturellement que nous avons introduit cet opérateur de Hill Climbing.

Choix d'un individu $\vec{x}_0$ dans la population
On se donne:
un nombre réel σ
un nombre maximum de tirages *NTMAX*
un nombre maximum de refus *NRMAX*

Début :
$\vec{x} \leftarrow \vec{x}_0$
$NR \leftarrow 0$
$NT \leftarrow 0$
Tant que ($NR < NRMAX$) **et** ($NT < NTMAX$) **faire**
Simuler une variable aléatoire $\vec{y}$ de loi $N(\vec{x}, \sigma Id)$
Si $f(\vec{y}) > f(\vec{x})$ **alors** $\vec{x} \leftarrow \vec{y}$ **et** $NR \leftarrow 0$ (*Amélioration*)
Si $f(\vec{y}) \leq f(\vec{x})$ **alors** $NR \leftarrow NR + 1$
$NT \leftarrow NT + 1$
fintq
fin

Figure II.4: Opérateur de recherche locale Gaussien

La figure II.5 présente l'algorithme génétique que nous emploierons dorénavant.

```
Program génétique

 /* Commentaires */

t          : génération courante
NBGEN      : génération maximum
NBPOP      : taille de la population
thau       : paramètre de pénalité
thau0      : valeur initiale de thau
thau_fin   : valeur finale de thau

 /* Fin commentaires */

début
 t = 0
 initialiser (P(t))
 évaluer (P(t))
 thau=thau0
 tant que (thau < thau_fin) faire
        tant que ( t < NBGEN ) faire
                P'(t)=sélection(P(t))
                tant que (dim(P(t)) < NBPOP ) faire
                        ioper=sélection_opérateur ( )
                        choisir aléatoirement 1 ou 2 individus
                        croiser ou muter cet ou ces individus à l'aide de
                        l'opérateur ioper
                        placer les descendants dans P(t+1)
                fintq
                évaluer (P(t+1))
                t = t + 1
        fintq
        augmenter (thau )
 fintq
fin
```

Figure II.5: Algorithme génétique pour le codage réel

3.2 Comparaison des performances avec celles du codage binaire

Les probabilités des différents opérateurs en début et en fin d'exécution sont laissés aux soins de l'utilisateur:

Px_1^i et Px_1^f : probabilité d'apparition du croisement **1 point**.

Px_2^i et Px_2^f : probabilité d'apparition du croisement **2 points**.

Px_u^i et Px_u^f : probabilité d'apparition du croisement **Uniforme**.

Px_a^i et Px_a^f : probabilité d'apparition du croisement **Arithmétique**.

Pm_u^i et Pm_u^f : probabilité d'apparition de la mutation **Uniforme**.

Pm_{nu}^i et Pm_{nu}^f : probabilité d'apparition de la mutation **Non Uniforme**.

Pm_b^i et Pm_b^f : probabilité d'apparition de la mutation **aux bornes**.

Pm_h^i et Pm_h^f : Probabilité d'apparition de l'opérateur de **Hill Climbing**.

La mutation dynamique dépend du paramètre ***b*** qui définit son degré de non uniformité et l'opérateur de Hill-Climbing des paramètres σ et **NTMAX** et **NRMAX** qui sont respectivement l'écart type de la gaussienne, le nombre maximum de tirages et le nombre maximum de refus consécutifs.

La première fonction que nous avons testée est la fonction de Rastrigin pour n=10, pour laquelle l'algorithme génétique binaire n'avait jamais localisé l'optimum ni même sa région. L'algorithme a été exécuté 30 fois. Les tables II.1 et II.2 montrent les valeurs des paramètres de l'algorithme que nous avons utilisées: les valeurs finales des probabilités sont atteintes lors de la dernière génération que nous avons fixée à 500. On peut noter que la somme de ces probabilités n'est pas égale à 1 ce qui veut dire que des individus ne sont pas affectés par ces opérateurs après l'étape de sélection. On constate aussi que les probabilités de Croisement sont bien plus importantes en début d'exécution qu'en fin d'exécution et qu'inversement, pour les mutations, les probabilités sont plus importantes à la fin. Les résultats obtenus (Table II.3) sont très convaincants puisque l'algorithme a localisé l'optimum dans 29 essais sur 30 et la précision de 3 chiffres significatifs après la virgule est atteinte pour 13146 évaluations en moyenne. Enfin, la région de l'optimum global est approché lorsque 8732 évaluations en moyenne ont été calculées.

Px_1^i	Px_2^i	Px_u^i	Px_a^i	Pm_u^i	Pm_{nu}^i	Pm_b^i	Pm_h^i
0.05	*0.05*	*0.05*	*0.25*	*0.05*	*0.05*	*0.003*	*0.0*
Px_1^f	Px_2^f	Px_u^f	Px_a^f	Pm_u^f	Pm_{nu}^f	Pm_b^f	Pm_h^f
0.01	*0.01*	*0.01*	*0.15*	*0.10*	*0.10*	*0.003*	*0.05*

Table II.1: Probabilités d'apparition des opérateurs en début et en fin d'exécution

NBPOP	*b*	NTMAX	NRMAX	tsel
70	*2.0*	*12*	*8*	*1.9*

Table II.2: Valeurs des autres paramètres de l'AG

Fonction	n	nopt	neval
f_4	*10*	*29*	*13146*

Table II.3: Performance de l'algorithme génétique sur la fonction de Rastrigin pour n=10

Cet exemple montre que les performances de l'algorithme sont grandement améliorées par l'utilisation du codage réel. Il prouve encore une fois la capacité de l'algorithme génétique à localiser l'optimum global. De plus, compte tenu du nombre très important d'optima locaux de cette fonction, on peut s'attendre à ce que l'algorithme génétique soit performant sur les problèmes réels en électromagnétisme où, d'après [DeVasconcelos94b], le nombre d'optima est bien moins important (au moins en optimisation de forme).
Nous avons également testé la fonction de Rastrigin pour n=5 pour pouvoir comparer la précision du

codage réel avec celui de l'algorithme binaire. Nous avons conservé les même valeurs des paramètres sauf la taille de la population que nous avons fixé à 60 et la valeur **tsel**=1,7. Sur les 30 essais réalisés l'optimum a été trouvé avec une précision de 3 chiffres significatifs après la virgule. Mais ce qu'on peut de plus constater (Table II.4), c'est que le nombre d'évaluations pour l'atteindre est nettement moins important que dans le cas du codage binaire. Nous avons aussi fait figurer le nombre d'évaluations nécessaire pour localiser la région de l'optimum et les conclusions vis à vis du codage binaire sont identiques. Le codage réel apporte donc un "plus" à la fois dans la capacité de recherche de l'optimum mais aussi en termes de précision sur l'optimum.

Fonction	n	nopt	Région	neval
f_4	5	30	4300	7801

Table II.4: Performance de l'algorithme génétique sur la fonction de Rastrigin pour n=5

Nous avons ensuite testé la fonction de Rosen-Susiki avec une taille de population de 50 et une valeur **tsel**=1,7 mais en conservant les mêmes valeurs de probabilités que pour la fonction de Rastrigin. Nous montrons dans la table II.5 que pour un nombre d'évaluations sensiblement équivalent à celui de l'algorithme binaire (cf. chapitre I) l'erreur relative sur la valeur de la fonction à l'optimum est, dans le cas réel, de 10^{-3} et non plus de 10^{-2}.

Fonction	n	nopt	Valeur à l'optimum	neval
f_5	4	30	(0.003, 0.999, 2.000, -0.999)	7585

Table II.5: Amélioration de la précision dans le cas réel sur la fonction de Rosen-Susiki

Une fois encore, l'utilisation du codage réel a permis d'améliorer la précision de la solution.
La table II.6 présente la progression du meilleur individu en fonction du nombre d'évaluations déjà réalisées lorsque l'algorithme génétique possède l'opérateur de Hill-Climbing *(2^{ere} colonne)* et lorsqu'il ne le possède pas *(3^{ieme} colonne)*. Les tests ont été fait sur la fonction de Rastrigin pour n=4 et les valeurs des différents paramètres sont montrées dans les tables II.7 et II.8. La valeur de σ est initialement fixée à 0.1 puis décroît à mesure que l'algorithme progresse.

neval	Valeur de f_4 avec Hill-Climbing	Valeur de f_4 sans Hill-Climbing
1000	2.30	2.50
2000	0.92	1.05
3000	0.38	0.46
4000	0.000085	0.070
5000	0.0000078	0.0096
6000	0.00000095	0.0028
7000	$<10^{-11}$	0.0000076

Table II.6: Performance de l'AG avec opérateur de mutation "Hill-Climbing"

Px_1^i	Px_2^i	Px_u^i	Px_a^i	Pm_u^i	Pm_{nu}^i	Pm_b^i	Pm_h^i
0.05	*0.05*	*0.05*	*0.25*	*0.05*	*0.05*	*0.003*	*0.0*
Px_1^f	Px_2^f	Px_u^f	Px_a^f	Pm_u^f	Pm_{nu}^f	Pm_b^f	Pm_h^f
0.01	*0.01*	*0.01*	*0.15*	*0.10*	*0.10*	*0.003*	***0.0 ou 0.05***

Table II.7: Probabilités d'apparition des opérateurs pour la fonction f_4

NBPOP	*b*	NTMAX	NRMAX	tsel
50	*2.0*	*12*	*8*	*1,7*

Table II.8: Valeurs des autres paramètres de l'AG

On constate que cet opérateur apporte un gain sur la précision de la solution puisqu'il escalade les pics (pour une maximisation !) de la région de l'individu auquel il est appliqué. Cet opérateur doit cependant être appliqué avec une probabilité assez faible pour éviter que la population ne converge vers le point "escaladé".

Pour terminer, il faut noter que nous avons conservé des probabilités d'apparition identiques pour tous les problèmes traités.

4 HYBRIDATION

L'optimum obtenu par une méthode déterministe est a priori local, sauf si bien sûr le point initial se trouve proche de la solution. Mais en général cette éventualité n'est que le fruit d'un heureux concours de circonstances. En fait l'intérêt des méthodes déterministes réside dans le faible nombre d'évaluations nécessaire pour atteindre l'optimum.
Les algorithmes stochastiques, que nous avons vus, possèdent deux avantages. Ils ne nécessitent pas le calcul des gradients et surtout ils sont capables de localiser la région de l'optimum global. Cependant, pour arriver à des solutions précises, il faut payer un prix : un nombre d'évaluations important.
Nous envisageons donc le couplage d'un algorithme génétique et d'une méthode déterministe pour tirer profit de leurs avantages en s'affranchissant (au moins partiellement) de leurs inconvénients. Notre algorithme génétique se charge alors de fournir un point initial pour l'algorithme déterministe qu'on espère être situé dans la région de l'optimum global. Ensuite, la méthode déterministe plonge rapidement et sans trop d'efforts de calcul vers l'optimum. Mais à quel moment l'algorithme génétique a t-il localisé la bonne région ? Il n'y a pas de réponses exactes puisque l'optimum n'est pas assuré ni dans la théorie - temps infini - et encore moins dans la pratique où on se fixe un nombre maximum d'évaluations. Cependant, certains critères de commutations peuvent être mis en œuvre. Le plus simple consiste à arrêter l'algorithme génétique lorsqu'un nombre maximum d'évaluations a été atteint. Pour déterminer ce nombre, il faudra s'appuyer sur la connaissance que l'on a des caractéristiques de la fonction. Intuitivement on sent bien qu'une fonction dans un espace de faible dimension et ne possédant que 2 ou 3 optima locaux, ne nécessitera pas autant d'exploration - et donc d'évaluations -

que la fonction de Rastrigin en dimension 10. Malheureusement, ce genre de connaissances n'est pas forcément accessible même lorsque la fonction à optimiser est donnée sous forme analytique ! On peut alors proposer des critères de commutation plus élaborés comme par exemple un critère basé sur les génotypes des individus. Dans ce cas, on mesure la convergence de l'algorithme en calculant le nombre de gènes ayant convergé. On entend par "gène convergé" un gène qui aurait la même valeur (ou une valeur proche dans le cas du codage réel) dans une majorité d'individus de la population. Une autre manière de vérifier la convergence de l'algorithme consiste à comparer l'adaptation entre les meilleurs individus de deux générations successives. L'algorithme commute alors quand la différence entre ces deux valeurs d'adaptation est inférieure à une certaine valeur très faible. Néanmoins, ces deux méthodes ne sont absolument pas fiables. Elles indiquent bien sûr une convergence de l'algorithme, mais elles n'assurent en aucun cas que cette convergence s'est produite à l'optimum. Notamment, il peut être intéressant de laisser continuer l'AG sur quelques générations car une petite mutation peut relancer la recherche et faire migrer les individus vers d'autres régions. De plus, si un super-individu est généré il y a de fortes chances pour qu'il soit le meilleur durant plusieurs générations : le deuxième critère ne nous assurerait donc même pas de la convergence de l'algorithme vers un optimum local ! Le choix du bon critère de commutation semble sans réponse. Pour notre algorithme hybride nous avons choisi un critère basé sur le génotype des individus. Il se présente de la manière suivante:

Pour un gène placé en position i on calcule les écarts relatifs $\frac{|x_i - x_i^{best}|}{|x_i^{best}|}$ où x_i représente la valeur du gène i pour un individu $\bar{x}$ quelconque de la population et x_i^{best} cette même valeur mais pour le meilleur individu déjà généré par l'algorithme. Si un certain nombre de ces écarts - calculé comme un pourcentage de la taille de la population - est inférieur à une valeur choisie faible, alors on considérera que le gêne en position i a convergé. Cette même opération est répétée sur tous les gènes et l'algorithme est arrêté lorsque tous les gènes ont convergé. Ce critère considère donc que l'algorithme a suffisamment cherché lorsque tous les individus ont migré vers la même région de l'espace. On va maintenant étudier de façon quantitative la viabilité d'un tel critère en testant notre algorithme hybride sur certaines fonctions.

Les paramètres pour mesurer la commutation sont:

cv^i : Paramètre qui définit la convergence du gène i : le gène i a convergé si $\frac{|x_i - x_i^{best}|}{|x_i^{best}|} < cv^i$

%pop : Pourcentage de la population ayant convergé vers le meilleur individu généré jusqu'alors par l'algorithme.

La première fonction testée est celle de Rastrigin pour n=5. Les valeurs des différents paramètres de l'algorithme sont celles utilisées précédemment pour cette même fonction. Les essais ont été répétés 30 fois. La table II.9 présente les résultats obtenus pour différentes valeurs de cv^i et **%pop**. Lorsque tout les gènes ont effectivement convergé, l'algorithme génétique s'arrête et fournit son meilleur individu à l'algorithme de gradient qui l'utilise alors comme point de départ de sa recherche. Pour comparer les

critères de commutation nous avons fait figurer le nombre d'essais ayant abouti à l'optimum et le nombre moyen d'évaluations qu'il a fallu à l'algorithme génétique avant de commuter. Nous avons vu plus haut que l'AG nécessitait environ 4300 évaluations pour localiser la région de l'optimum sur cette fonction; l'idéal serait donc que la commutation se produise lorsque ce nombre d'évaluations est atteint. Cependant, il faut bien noter que si la région est bien découverte il n'est en revanche pas certain que la population ait convergé vers cette région.

On constate que si cv^i est trop "faible", les gènes convergent rapidement (1500 évaluations) vers une région qui n'est pas celle de l'optimum global, surtout que pour ce nombre d'évaluations les probabilités de mutation ne sont pas trop élevées (20^{ieme} génération). A contrario, dans le dernier test ($cv^i=10^{-5}$ **%pop**=97) tous les essais ont conduit à l'optimum mais la commutation ne s'est jamais effectuée si bien que la population a continué durant 400 générations et l'algorithme a donc réalisé près de 12 000 évaluations. Entre ces deux valeurs extrêmes c'est le 7ieme test qui a donné les meilleurs résultats: environ 7000 évaluations sont faites par l'AG avant de commuter. La table II.10 montre la solution obtenue au moment de la commutation, la valeur de l'objectif et le nombre d'évaluations de l'algorithme de gradient pour atteindre l'optimum avec une erreur relative de 10^{-12} sur les variables.

cv^i	%pop	nopt	neval
10^{-2}	90	0	1500
10^{-3}	90	12	3900
10^{-3}	95	11	3012
10^{-4}	90	19	4803
10^{-4}	95	21	5421
10^{-5}	90	22	5736
10^{-5}	95	27	7313
10^{-5}	95	30	pas de commutation

Table II.9: Commutations sur la fonction de Rastrigin pour n=5

neval (AG)	Meilleur individu au moment de la commutation	neval (BFGS)	neval (Total)
7313	(2.5003, 2.4997, 2.49952, 2.5006, 2,5003)	100+500	7913

Table II.10: Nombre d'évaluations moyen pour atteindre l'optimum dans le cas hybride pour la fonction de Rastrigin avec n=5

Les autres tests montrent cependant qu'il peut être intéressant de laisser "tourner" l'algorithme génétique un certain nombre de générations supplémentaires. Notamment, le critère de la convergence de 90% de la population n'implique pas forcément que l'algorithme ait terminé de chercher: les autres

individus et des mutations de plus en plus fréquentes sont des sources d'informations très importantes. En effet dans tous les cas l'algorithme génétique finit par localiser l'optimum si on ne fait pas de commutation (cf. Table II.4).

Il faut cependant noter que lorsque l'optimum global n'a pas été trouvé, la solution obtenue par la méthode hybride était le 2ieme optimum (sauf pour les 3 premiers tests): on voit donc (Table II.9) que pour obtenir avec une très bonne précision le premier ou le second optimum, la méthode hybride nécessite entre 4803 et 7313 évaluations de la fonction par l'algorithme génétique (auxquelles il faut ajouter environ 400 évaluations pour la méthode de gradient).

Finalement nous avons étudié le comportement de ce critère de commutation sur la fonction $f7$ (Fig. II.6):

$$Min f_7(\boldsymbol{x}) = \left(x_1^2 + x_2^2\right)^{0.25}\left(sin^2\left(x_1^2 + x_2^2\right)^{0.1} + 1\right) = f(0,0,\ldots,0) = 0$$

avec $-100. \leq x_i \leq 100$

Outre le grand nombre d'optima, la difficulté de cette fonction provient du fait que la région de l'optimum global est très étroite (Fig. II.7).

L'algorithme génétique localise l'optimum en peu d'évaluations (environ 4000) mais si on utilise le critère précédent (cv^i=10^{-5} **%pop**=95) alors la commutation ne se produit pas et l'algorithme continue de chercher jusqu'à la dernière génération (400).

Cet exemple montre donc que le choix du bon critère est assez difficile à obtenir et apparemment dépend bien du problème étudié. Si on se réfère à ces deux exemples, on pourra a priori utiliser ce critère mais en ajoutant une condition sur le nombre d'évaluations qu'il ne faudra pas dépasser.

La méthode hybride permet quand même d'obtenir une solution avec une bien meilleure précision que dans le cas d'un algorithme génétique utilisé seul comme on peut le voir dans la table II.11. Pour la fonction f_7, l'algorithme est arrêté après 10 000 évaluations et fournit son meilleur individu à la méthode de gradient.

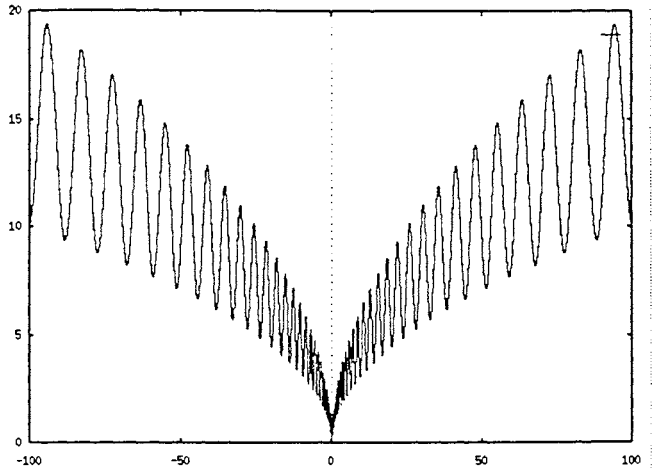


Figure II.6 : Fonction f7 sur une dimension

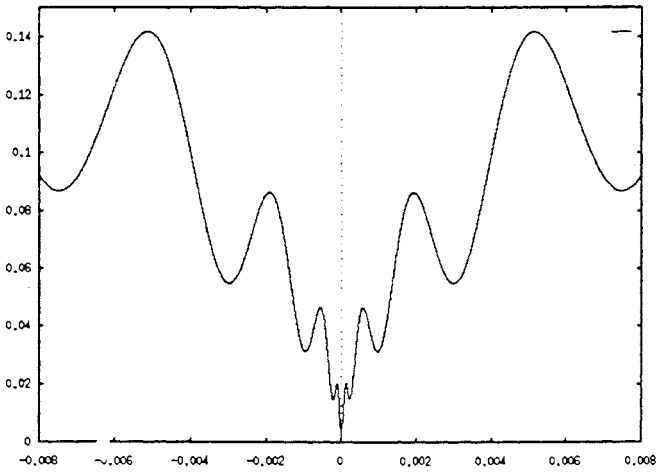


Figure II.7: Fonction f7 près de l'optimum

Meilleur individu lors de l'arrêt de l'AG	Solution obtenue	neval (BFGS)
$(5.9\ 10^{-7}, 5.7\ 10^{-10})$	$(10^{-14}, 9{,}8^{-15})$	60+120

Table II.11: Précision obtenue sur la fonction f_7

5 PARALLELISATION

Nous venons de voir que l'hybridation constituait un moyen pour réduire les efforts de calcul. Cependant, la localisation de la région de l'optimum nécessite tout de même un nombre important d'évaluations.

L'utilisation d'ordinateurs multiprocesseurs est une alternative efficace pour réduire les temps d'exécution. Il n'y a qu'à imaginer qu'une moitié des évaluations soit faite sur un processeur et l'autre moitié sur un autre processeur: on diviserait *théoriquement* le temps de calcul par 2. En fait, l'utilisation d'architectures parallèles a donné lieu à de nouvelles implantations des algorithmes

génétiques [Pettey87][Pettey89][Tanese89][Elketroussi94]. Dans ces implantations, il ne s'agit pas simplement de distribuer le travail d'évaluation mais aussi de faire évoluer des populations de manière différentes. Ces populations peuvent communiquer entre elles en s'échangeant, par exemple, des individus. Nous présenterons ici l'algorithme PGA (Parallel Genetic Algorithm) en guise d'illustration car le parallélisme semble être une voie prometteuse pour les algorithmes génétiques. Dans l'algorithme PGA, chaque processeur traite une sous-population SP_i sur K générations à l'aide d'un algorithme génétique séquentiel. Ensuite, une phase de communication entre les différents processeurs est réalisée et les différentes sous-populations repartent pour K nouvelles générations (Fig. II.8).
[Pettey87] a proposé un algorithme parallèle dans lequel la phase de communication se produit à chaque génération des algorithmes séquentiels (K=1). Cette communication consiste, pour chaque sous population, à envoyer son meilleur individu aux populations voisines et à recevoir de celles-ci leurs meilleurs individus. Pour insérer ces nouveaux entrants, Pettey propose de remplacer les moins bons individus ou des individus choisis aléatoirement.

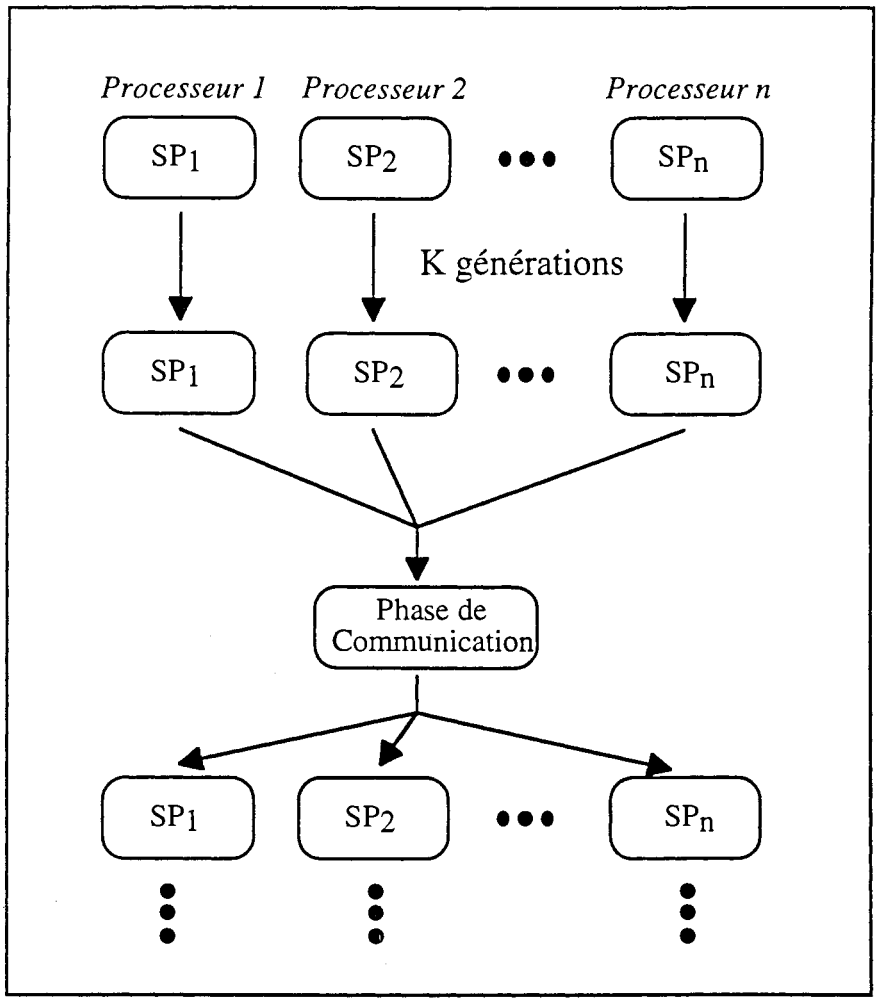


Figure II.8: Schéma de principe d'un algorithme génétique parallèle

[Elketroussi94] propose un schéma plus complet. Chaque population évolue indépendamment pendant K générations. Le but est d'avoir dans chaque sous-population une distribution des individus autour

d'un optimum (local a priori à la convergence). Ensuite, si, à l'intérieur d'une sous population SP_i, les 2 meilleurs individus ont convergé vers 2 zones distinctes alors ils sont placés dans 2 nouvelles sous populations SP_{i1} et SP_{i2} à la prochaine étape de l'algorithme. Par contre si deux sous-populations ont convergé vers le même optimum alors celles-ci ne formeront plus qu'une population à la prochaine étape.
Par rapport aux algorithmes génétiques traditionnels, ces implantations parallèles permettent l'échange d'informations non plus entre les seuls individus d'une même population mais aussi entre les populations.
[Mühlenbein91] a étudié les algorithmes parallèles et a montré que non seulement ces algorithmes permettaient un gain en temps cpu mais qu'en plus ils permettaient de résoudre des problèmes qui n'avaient pas pu l'être par l'algorithme séquentiel.

Il reste cependant encore beaucoup de possibilités à explorer avec le parallélisme. Notamment sur les stratégies de recherche utilisées dans chacune des populations ou sur les phases de communications entre ces populations.
L'algorithme parallèle a été présenté uniquement pour montrer les possibilités qui s'offrent à l'utilisateur dans sa recherche d'un algorithme à la fois performant et peu coûteux en temps de calcul.

6 CONCLUSION

Dans ce chapitre nous nous sommes interrogés sur l'intérêt du codage binaire et sur les difficultés qui peuvent apparaître lorsque les structures manipulées n'étaient pas des paramètres réels. Notamment, ne vaut t-il pas mieux conserver le codage naturel du problème et modifier les opérateurs de la recherche en fonction de ce codage? Le codage "réel" des paramètres réels a donné sur les exemples traités de meilleurs résultats que le codage binaire:

- il a localisé l'optimum global avec un nombre moins important d'évaluations.
- il a amélioré la précision obtenue sur la solution.

De plus, ce type de codage a autorisé l'utilisation d'opérateurs de nature plus variée répondant à deux principes très simples:

1) La *mutation* est un opérateur qui perturbe l'information contenue dans un individu.
2) Le *croisement* est un opérateur qui échange de l'information entre plusieurs individus.

Le nombre plus important d'opérateurs augmente cependant le nombre de paramètres de l'algorithme à régler, comme les fréquences d'apparition des opérateurs par exemple. Dans notre algorithme nous avons choisi une Roue de Loterie déjà proposé par Davis pour choisir entre l'opérateur de croisement et l'opérateur de mutation dans l'algorithme binaire, et nous l'avons adapté à nos opérateurs. Notamment, nous avons mis à jour les fréquences d'apparition des opérateurs à l'aide de considérations empiriques comme "plus de mutations en fin d'exécution" et "plus de croisements au début". Ce choix nous a permis d'éviter de mettre en œuvre des structures très lourdes servant à récompenser les opérateurs travaillant le mieux lors de la recherche tout en assurant la convergence vers l'optimum global.

Nous avons aussi étudié l'hybridation entre l'algorithme génétique et un algorithme de type gradient. La difficulté est de trouver le "bon moment" pour commuter d'un algorithme vers l'autre. Pour cela, nous avons mis en œuvre un critère basé sur le génotype des individus de la population. Ce critère force la population à converger vers le meilleur individu déjà généré par l'algorithme avant de commuter. Il a donné de bons résultats sur la première fonction testée mais a laissé tourner l'algorithme plus de temps qu'il ne fallait pour localiser la région de l'optimum global sur le deuxième exemple. Ceci semble indiquer que nous n'avons pas trouvé de critère de commutation réellement indépendant du problème traité. En revanche, l'algorithme de gradient a toujours nettement amélioré la précision de la solution obtenue par l'algorithme génétique. Dans notre implantation, nous avons conservé ce critère mais nous avons rajouté un nombre maximum d'évaluations qu'il ne faut pas dépasser.

Enfin, nous avons brièvement présenté ce qui semble être une voie très prometteuse pour l'optimisation génétique: l'algorithme génétique parallèle. Celui ci ne consiste pas seulement à partager le calcul des différentes évaluations entre les différents processeurs mais aussi à échanger des informations non plus seulement entre individus mais entre populations entières.

Chapitre III

Optimisation d'un refroidisseur de puce IGBT

Chapitre III

Optimisation d'un refroidisseur de puce IGBT

1 INTRODUCTION

Le travail qui suit est le fruit d'une collaboration avec l'équipe Electronique de Puissance du Laboratoire et en particulier avec M. Luc Meysenc, M. Stéphane Raël et M. Christian Schaeffer qui ont passé de longues heures à m'initier au composant IGBT ainsi qu'aux phénomènes thermiques que nous avons étudiés.
Ce travail, qui a réuni deux domaines a priori très éloignés, a débouché sur la conception d'un radiateur de dissipation thermique de grande performance pour puce IGBT.

L'IGBT est devenu le composant majeur de l'Electronique de Puissance car même s'il possède encore des temps de commutation supérieurs à ceux du MOS, il permet de contrôler des tensions beaucoup plus importantes.
En contrepartie, ces composants sont le siège de fortes puissances, dissipées pendant les phases de commutation et de conduction et sont donc soumis à de forts gradients de température. En utilisant des réseaux de canaux de taille microscopique parcourus par un fluide et placés sous la puce on peut maintenant, tout en limitant l'encombrement, évacuer des densités de puissance élevées avec une faible élévation de température. Cependant, des améliorations peuvent être apportées sur ce type de refroidisseur et dans ce chapitre nous proposerons une méthode originale basée sur les algorithmes génétiques pour dimensionner les micro-canaux de façon à obtenir un refroidissement optimal.
Dans ce qui suit nous ne détaillerons pas le comportement complet de l'IGBT et nous analyserons seulement ses caractéristiques d'un point de vue thermique.

2 TECHNOLOGIE DES MICRO-CANAUX

A cause des fortes puissances dissipées par les composants d'électronique de puissance il est indispensable d'utiliser un système de refroidissement.
Longtemps, ce refroidissement a été assuré par des radiateurs à convection d'air qui présentaient l'avantage d'une réalisation assez aisée. Malheureusement, l'éloignement entre le composant et le radiateur dans ce type d'architecture pénalise les performances thermiques de l'ensemble en raison du nombre de couches de matériaux qu'il faut ajouter ainsi que du grand nombre d'interfaces de contacts. Alors, pour répondre convenablement au besoins croissants de caractéristiques thermiques de plus en plus élevées pour les composants d'électronique de puissance, la séparation entre les organes électrique et thermique a été supprimée. Ainsi, en utilisant des micro-canaux placés directement sous la puce et parcouru par un fluide incompressible on autorise l'évacuation de quantité plus importante de chaleur et de plus, on limite considérablement l'encombrement.
Pour intégrer ce micro-échangeur, on peut par exemple braser la puce sur un support en cuivre dans lequel seront usinés des micro-canaux. Le flux de chaleur dissipé par le composant traverse alors la partie supérieure de l'échangeur pour être finalement évacué par le fluide (Fig. III.1).

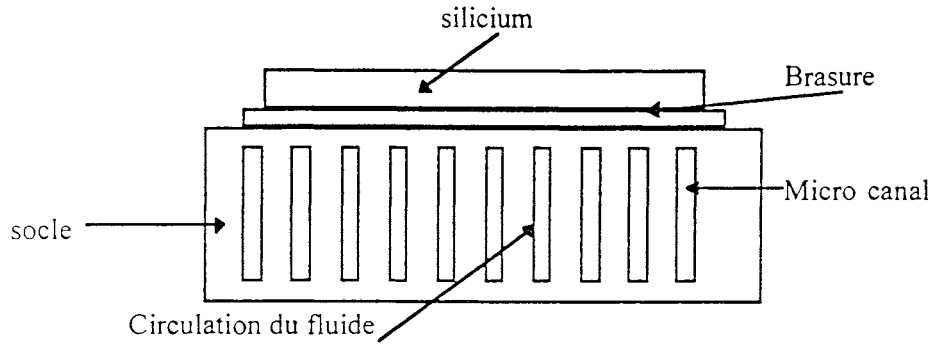


*Figure III.1: Micro-échangeur**

Il faut cependant noter que cette intégration directe a pour effet de porter au potentiel de drain tout le dispositif qui, dans le cas de l'IGBT, peut être très important (de l'ordre de 1000Volts). Il est donc nécessaire de limiter au maximum le volume soumis à ce potentiel. Ceci peut être réalisé de deux manières: la première consiste simplement à placer une couche isolante entre la brasure et l'échangeur, mais dans ce cas les propriétés thermiques du matériau risque de dégrader les performances thermiques de l'ensemble. On lui préfère donc une seconde technique qui consiste à utiliser un fluide caloporteur diélectrique comme le fluorinert ou l'eau désionisée (Fig. III.2).

* Dessins reproduits avec autorisation de M. Luc Meysenc

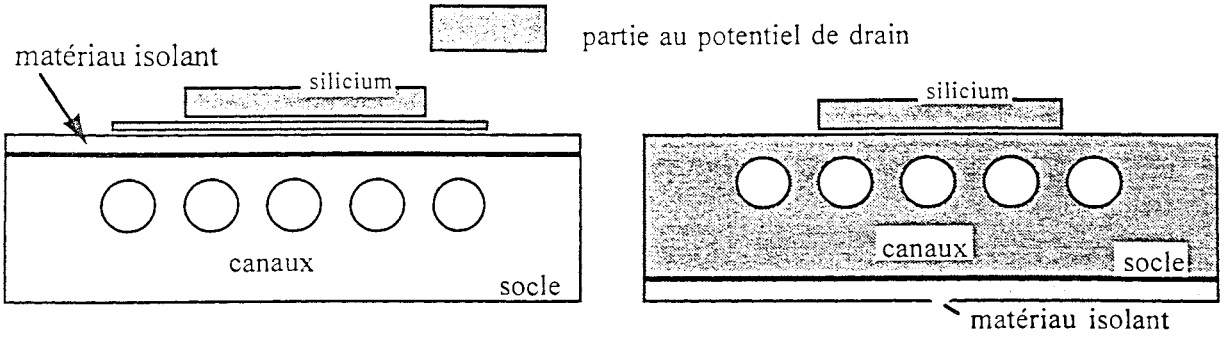


*Figure III.2: Isolation électrique de l'ensemble**

3 ETUDE THERMIQUE DU DISPOSITIF

L'ensemble du dispositif peut être décrit plus précisément par un ensemble de paramètres (Fig. III.3). La puce est brasée sur un parallélépipède en cuivre dans lequel sont usinés des micro-canaux. Le fluide incompressible se propage le long des canaux dans la direction z.
La chaleur est évacuée par la puce vers les ailettes (d'épaisseur e) puis vers le fluide. Les paramètres W et L sont imposés par la taille de la puce et la hauteur d doit être aussi faible que possible pour limiter l'écart de température résultant de la conduction dans le matériau. Cette grandeur dépend entre autres de la manière dont est réalisé la brasure. Finalement l'échangeur peut être décrit par la donnée de D, e, lc qui représentent respectivement la profondeur du canal, la largeur du canal et la largeur des ailettes.

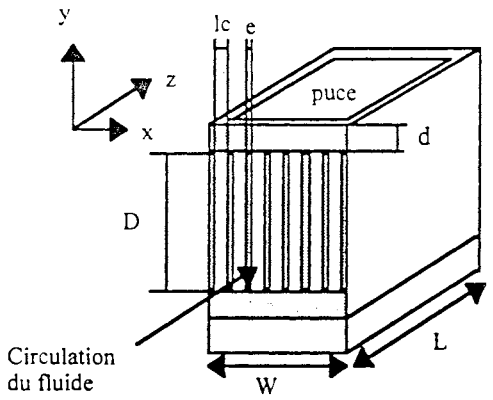


Figure III.3: Paramètres de l'échangeur

L'analyse thermique met en évidence deux gradients de température (Fig. III.4). Le premier est dû à l'échange convectif entre le fluide et la paroi de l'échangeur et peut se modéliser par une résistance thermique Rth_{conv}. Le second est dû à l'élévation de température du fluide à mesure qu'il absorbe l'énergie dissipée par le composant et on le modélise par une résistance thermique Rth_{cap} [Meysenc96a]. La résistance thermique totale est alors égale à la somme $Rth_{conv}+Rth_{cap}$ et elle rend compte de l'écart de température entre la température du fluide en entrée et la température de la paroi en

* Dessins reproduits avec autorisation de M. Luc Meysenc

sortie des canaux. Si de plus on néglige les conductions thermiques au niveau de la brasure alors on peut définir la température de jonction comme étant la température de la paroi à la sortie d'un canal.

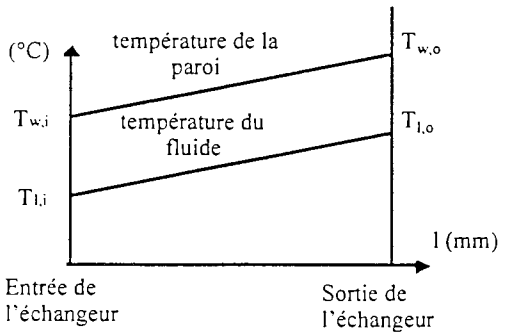


Figure III.4: Profils de température

La résistance thermique Rth_{cap} peut être exprimée de façon classique par:

$$Rth_{cap} = \frac{1}{Q\rho_l Cp_l} \qquad \text{(III.1)}$$

où Q désigne le débit du fluide (m^3s^{-1}), ρ_l la masse volumique du fluide et Cp_l la chaleur spécifique du fluide.

En revanche, l'expression de Rth_{conv} est plus difficile à calculer car elle fait intervenir un coefficient d'échange thermique qui dépend des conditions expérimentales dans lesquelles on se place et en particulier:

- des caractéristiques géométriques de l'écoulement (forme, rugosité).
- des propriétés du fluide.
- du type d'écoulement (laminaire, turbulent).

[Tuckerman85] a calculé ce coefficient à l'aide de l'expression du nombre de Nusselt qu'il a tiré de corrélations sur des résultats expérimentaux. Cependant, son expression n'est valable qu'en régime laminaire. Nous avons donc utilisé les corrélations de [Knight92] qui ont l'avantage d'être utilisables en régime laminaire et turbulent et qui sont de plus, bien adaptées à la géométrie et à la taille des canaux de notre application [Meysenc96b].
Cette méthode de calcul du coefficient d'échange se base sur trois grandeurs classiques sans dimension:
Le nombre de Reynolds qui représente le rapport des forces d'inertie aux forces de frottement et qui s'exprime par:

$$Re = \frac{\rho_l V D_h}{\mu_l} \qquad \text{(III.2)}$$

où V désigne la vitesse du fluide (ms^{-1}), μ_l la viscosité cinématique du fluide ($kgs^{-1}m^{-1}$) et D_h le diamètre hydraulique donné par la formule:

$$D_h = \frac{2\,D\,lc}{D + lc} \tag{III.3}$$

et le nombre de Prandtl défini par:

$$Pr = \frac{\mu_l\, Cp_l}{k_l} \tag{III.4}$$

où k_l est la conductivité thermique du fluide ($W\,°C^{-1}m^{-1}$).
Enfin nous rajoutons un paramètre géométrique sans dimension:

$$G = \frac{\left(\frac{lc}{D}\right)^2 + 1}{\left(\frac{lc}{D} + 1\right)^2} \tag{III.5}$$

A l'aide de ces grandeurs caractéristiques, on peut définir le nombre de Nusselt et le coefficient de frottement en régime laminaire et en régime turbulent. Ainsi, si le nombre de Reynolds est inférieur à 2300 alors l'écoulement est laminaire et si on note respectivement Nu et f le nombre de Nusselt et le coefficient de frottement on a:

$$Nu = -1.047 + 9.326\,G \tag{III.6}$$

$$f = \frac{4.7 + 19.64\,G}{Re} \tag{III.7}$$

Si le nombre de Reynolds est supérieur ou égal à 2300 alors Nu et f deviennent:

$$Nu = \frac{\frac{f}{2}(Re - 100)Pr}{1 + 12.7\sqrt{\frac{f}{2}}\left(Pr^{2/3} - 1\right)} \tag{III.8}$$

$$f = \left(0.0929 + \frac{1.01612\,D_h}{L}\right) Re^{\left(-0.268 - \frac{0.3193\,D_h}{L}\right)} \tag{III.9}$$

Finalement, à partir du nombre de Nusselt on déduit le coefficient d'échange h ainsi que l'expression théorique de la résistance thermique $Rth_{conv.}$

$$h = \frac{k_1 Nu}{D_h} \tag{III.10}$$

$$Rth_{conv} = \frac{1}{2nhL(D+lc)} \tag{III.11}$$

En général, on introduit la notion d'efficacité d'ailette pour tenir compte de la répartition de la densité de flux le long de la paroi des canaux. En effet, la partie la plus éloignée de la puce dissipe moins que la partie la plus proche et on corrige donc (III.11) à l'aide d'un facteur multiplicatif β défini comme suit:

$$\beta = \frac{tanh(mD)}{mD} \tag{III.12}$$

$$\text{avec } m = \sqrt{\frac{2h}{ek_p}} \tag{III.13}$$

et k_p la conductivité thermique de la paroi.

ce qui donne pour la résistance thermique:

$$Rth_{conv} = \frac{1}{2nhL(D+lc)\beta} \tag{III.14}$$

Nous venons donc de décrire les phénomènes thermiques qui se produisent dans le dispositif puce-échangeur. Ces phénomènes peuvent être modélisés par une résistance thermique:

$$Rth = Rth_{conv} + Rth_{cap} \tag{III.15}$$

Nous désirons maintenant minimiser cette Rth en jouant sur la taille et quelques autres paramètres de l'échangeur afin d'assurer un refroidissement optimum du composant de puissance (dans notre étude l'IGBT). Il faut donc définir précisément les paramètres qui sont susceptibles d'être modifiés ainsi que les contraintes de réalisation.

4 MINIMISATION DE LA RÉSISTANCE THERMIQUE TOTALE

Les données de notre problème sont les dimensions de la puce, les propriétés thermiques de l'ensemble des constituants et les propriétés du fluide employé. Tous ces paramètres sont donc fixes et les seules grandeurs sur lesquelles nous réaliserons notre optimisation sont :

- la profondeur des canaux (D)
- la largeur des canaux (lc)
- la largeur d'une ailette (e)
- la vitesse d'écoulement du fluide (V)

A l'aide des formulations précédentes on peut donc définir une résistance thermique qui soit fonction de ces quatre paramètres. Outre les contraintes de domaines sur ces quatre paramètres, il faut ajouter des contraintes représentatives du coût investi dans le système de pompage. Nous avons à cet effet retenu deux grandeurs caractéristiques:

- la chute de pression ΔP dans le micro-échangeur (Pa)
- la puissance motrice Pm assurant la circulation du fluide (W)

Finalement nous pouvons formaliser notre problème d'optimisation comme suit:

$$\begin{cases} Min \quad Rth(D,e,lc,V) \\ \\ D \in [D_{min}, D_{max}] \\ e \in [e_{min}, e_{max}] \\ lc \in [lc_{min}, lc_{max}] \\ V \in [V_{min}, V_{max}] \\ \\ \Delta P \leq \Delta P_{max} \\ Pm \leq Pm_{max} \end{cases} \qquad \text{(III.16)}$$

Les contraintes ΔP et Pm ne sont pas indépendantes car la deuxième s'exprime à l'aide de la première par la relation:

$$Pm = \Delta P\, Q \qquad \text{(III.17)}$$

où Q désigne le débit volumique total.

$$Q = D\, lc\, V \frac{W}{lc+e} \qquad \text{(III.18)}$$

De plus on peut relier ΔP au coefficient de frottement:

$$\Delta P = \frac{2\rho_l V^2 f L}{D_h} \qquad \text{(III.19)}$$

La transition laminaire - turbulent rend le problème discontinu et ne permet donc pas l'utilisation de méthodes basées sur le calcul des dérivées. [Knight92] et [Tuckerman85] ont proposé des optimisations se basant essentiellement sur un balayage du domaine des solutions. Nous avons résumé les résultats obtenus dans la table III.1 pour certaines valeurs de la chute de pression et de la puissance motrice.

Cependant ces résultats sont obtenus pour des valeurs bien particulières des contraintes et de plus rien ne nous garantit qu'ils sont effectivement bien optimaux. C'est pour cette raison que nous avons utilisé un algorithme génétique.

auteurs	Tuckerman	Knight
D (µm)	365	365
e (µm)	57	81
lc (µm)	57	377
V ($m\ s^{-1}$)	6.	19.68
ΔP (Bar)	2.068	2.068
Pm (W)	2	11
Rth (°K/kW)	**64**	**56**

Table III.1: Résultats de la littérature (W=10 mm).

Dans les premiers essais, nous avons cherché à savoir si les valeurs obtenues par [Tuckerman85] et [Knight92] étaient bien optimales. Pour cela, nous nous sommes basés sur le fait que la contrainte en puissance était celle qui limitait le processus d'optimisation c'est-à-dire qu'à l'optimum trouvé par l'algorithme génétique, la valeur Pm_{max} était souvent atteinte. Nous avons donc fixé Pm_{max} aux valeurs obtenues par Tuckerman (2 W) et Knight (11 W), nous avons imposé les contraintes de domaines comme indiqué sur la table III.2 et finalement, nous avons laissé "tourner" l'algorithme. Les résultats obtenus sont présentés dans la table III.3. On constate que pour une même valeur de Pm et pour des chutes de pression identiques ou plus faibles, on obtient une résistance thermique très inférieure.

	Min.	**Max.**
D (µm)	100	1000
e (µm)	50	1000
lc (µm)	50	1000
V ($m\ s^{-1}$)	1	20

Table III.2: Contraintes de domaines sur les paramètres

	AG	**AG**
D (µm)	597	600
lc (µm)	50	92
e (µm)	50	51
V ($m\ s^{-1}$)	4.48	9.6
ΔP (Bar)	1.5	2
Pm (W)	2	11
Rth (°K/kW)	**48.4**	**38.4**

Table III.3: Résultats de l'Algorithme Génétique (W=10 mm)

L'autre intérêt de l'algorithme génétique réside dans son caractère automatique c'est à dire qu'on peut facilement imposer de nouvelles contraintes sur les paramètres: par exemple, si pour l'usinage des

micro-canaux, la fraise utilisée est de 100 µm alors lc devra être fixé à cette valeur. L'algorithme génétique donne de nouvelles valeurs optimales (table III.4 et III.5 pour différentes valeurs de la puissance Pm et de lc).

	AG	**AG**	**AG**
D (µm)	922	1000	1000
lc (µm)	**100**	**100**	**100.**
e (µm)	105	57.4	57.2
V (m s^{-1})	10.82	8.32	6.14
ΔP (Bar)	1,7	0.7	0.5
Pm (W)	**8**	**4**	**2**
Rth (°K/kW)	**37.2**	**66.9**	**68.5**

Table III.4: Tailles optimales des micro-canaux lorsque le paramètre lc est fixé à 100 µm (contrainte d'usinage)

	AG	**AG**	**AG**
D (µm)	1000	1000	1000
lc (µm)	**200**	**200**	**200.**
e (µm)	194	126	166
V (m s^{-1})	13.21	7.99	7.74
ΔP (Bar)	1,2	0.5	0.5
Pm (W)	**8**	**2.5**	**2**
Rth (°K/kW)	**40.8**	**55.2**	**68.5**

Table III.5: Tailles optimales des micro-canaux lorsque le paramètre lc est fixé à 200 µm (contrainte d'usinage)

5 VALIDATION EXPÉRIMENTALE

Le calcul de la résistance thermique présenté précédemment dépendait en grande partie des corrélations empiriques utilisées pour le calcul du nombre de Nusselt. Même si celles ci sont reconnues dans la littérature pour être valable dans le cas que nous avons étudié, il est impératif de procéder à des essais expérimentaux afin de les vérifier. Cette étude a été menée au laboratoire par M. Luc Meysenc dont nous allons présenter les conclusions et quelques photographies sur la réalisation du prototype.

Pour valider les corrélations, 3 prototypes ont été mis au point:

- Un prototype à canaux carrés permettant de valider les corrélations en régime turbulent.
- Un prototype à canaux peu profonds (donc peu sensible aux effets d'ailettes décrit par β) et étroits ce qui assure un écoulement du fluide laminaire. Ce prototype permettra alors de valider les corrélations en régime laminaire.
- Enfin un prototype à canaux profonds où l'effet d'ailette ne peut être négligé: il permettra de valider le facteur β utilisé dans les corrélations.

Concernant le premier prototype, les corrélations employées ont données de bons résultats avec des erreurs sur la résistance thermique, entre valeur calculée et mesurée, de l'ordre de 10%.

Les deux autres prototypes ont donné des résultats similaires entre valeur calculée et mesurée de la résistance thermique (Fig. III.5 pour le troisième prototype). Ces constats valident d'une part les corrélations en régime laminaire et d'autre part le coefficient d'efficacité d'ailette β utilisé.
Il faut noter que dans les deux cas le régime d'écoulement est laminaire et donc les longueurs d'établissement hydraulique et thermique ne sont plus négligeables (Fig. III.6). Ce qui explique que pour de forts débits (Fig. III.5), pour lesquels ces longueurs sont plus importantes, l'écart entre les mesures et les valeurs fournies par les corrélations (du régime établit) croissent. En négligeant ces longueurs d'établissements, les modèles utilisés en régimes établis permettent dans tous les cas de prédire la résistance thermique avec des erreurs inférieures à 20% ce qui est acceptable pour cette application.

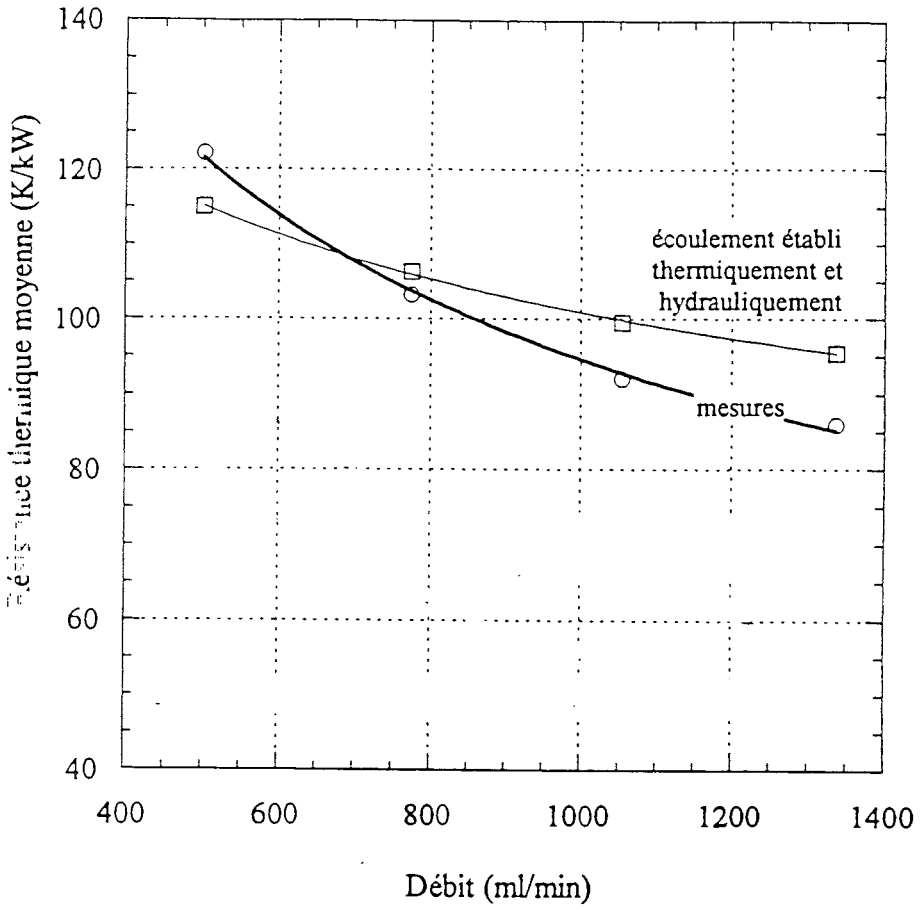


Figure III.5: Comparaison entre Rth expérimentale et corrélée dans le cas d'un régime d'écoulement laminaire établi

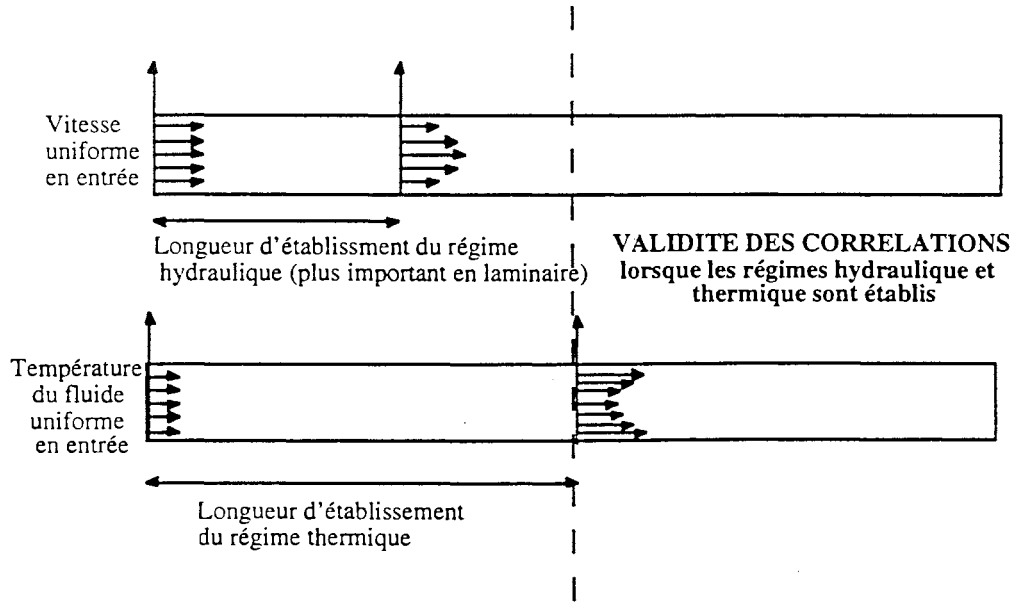


Figure III.6: Longueurs d'établissements des régimes hydraulique et thermique

Suivent quelques photographies (Fig. III.7, Fig. III.8) d'un des prototypes mis au point par M. Luc Meysenc.

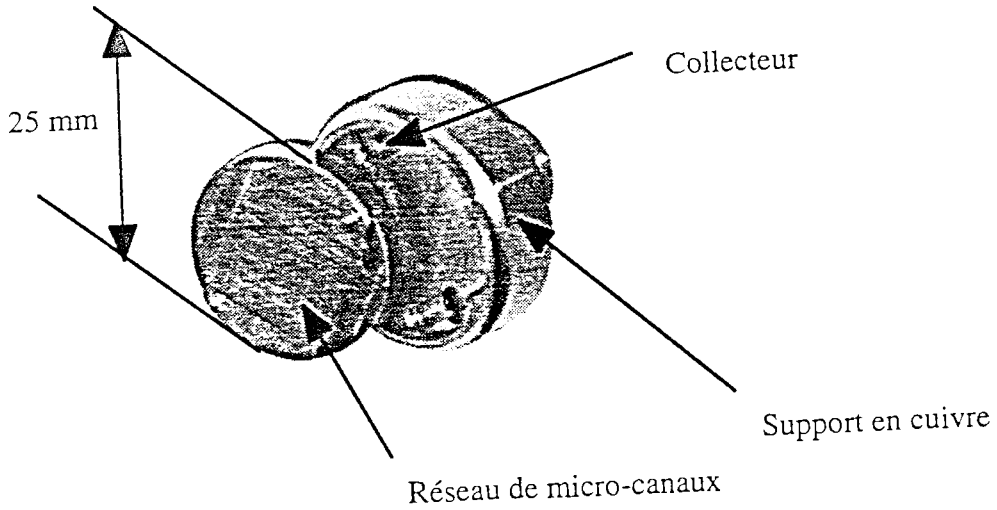


Figure III.7: Réseau de canaux dissocié de son support

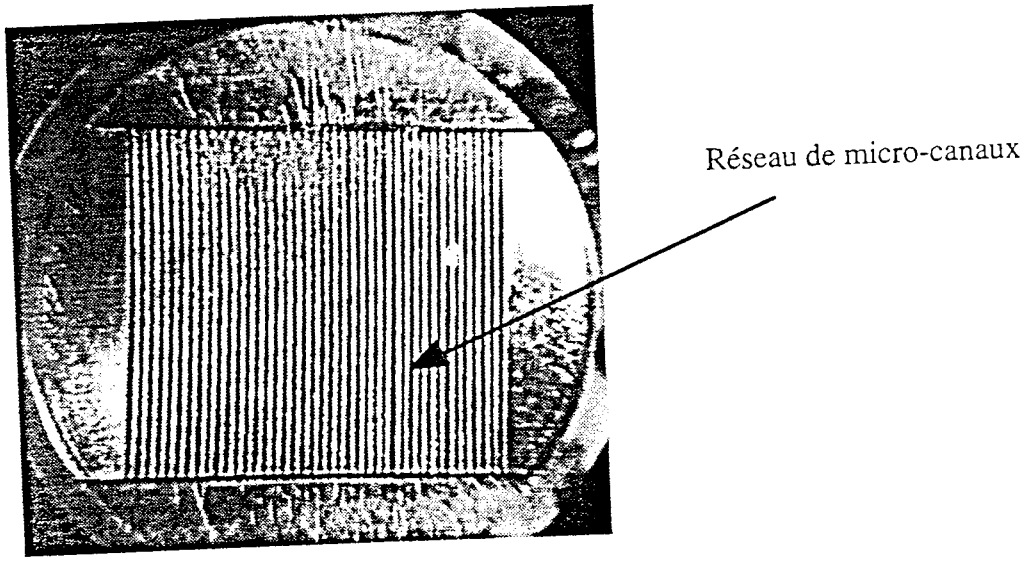


Figure III.8: Réseau de micro-canaux

6 CONCLUSION

Les besoins croissants de composants de puissances comme les IGBT ont conduit les ingénieurs à chercher des systèmes de refroidissement de plus en plus performants afin d'évacuer les fortes quantité de chaleur dissipées. La technologie de micro-canaux brasés directement sous la puce et parcourus par un fluide incompressible, répond à cette recherche et nous nous sommes alors attaché à optimiser leurs formes dans le but d'améliorer encore leurs performances.

Pour cela, nous avons défini une fonction objectif, établie en partie à l'aide de corrélations empiriques (vérifiées par l'expérience), ainsi que des contraintes représentatives du coût investit dans le système de pompage.

Cette fonction, la résistance thermique totale, a la particularité d'être discontinue au moins en un point situé à l'interface entre régime d'écoulement turbulent et régime d'écoulement laminaire du fluide, et comme de plus, rien ne nous indique à priori que cette fonction possède un unique optimum, c'est tout naturellement que nous avons utilisé notre algorithme génétique. Les résultats ont été très probant et la résistance thermique a pu être abaissée de façon sensible par rapport aux optimisations faites précédemment par certains auteurs.

Cette collaboration aura aussi eu le mérite de faire se rencontrer deux domaines de recherche possédant a priori peu de points communs ... et pourtant !

Chapitre V

Optimisation de forme de structures électromagnétiques

Chapitre IV

Optimisation de maillage CAO

1 INTRODUCTION ET DEFINITION DU PROBLEME

Les phénomènes électromagnétiques sont régis par des équations aux dérivées partielles (les équations de Maxwell) qui n'ont pas de solution analytique pour des problèmes complexes. La recherche d'une solution nécessite alors l'emploi d'une méthode numérique qui transforme le problème continu en un problème discret.

Dans la méthode *éléments finis* [Sabonnadière86] [Touzot84], le domaine étudié est découpé en sous domaines (les éléments finis) de formes géométriques simples (tétraèdres, hexaèdres, ...). Le problème est d'abord résolu sur ces sous domaines qui sont plus simples à traiter. Puis, la variable d'état (potentiel vecteur, potentiel scalaire ...), dont on cherche une solution sur tout le domaine, est reconstruite en assemblant les solutions obtenues sur les sous domaines. En fait, cette méthode ramène la résolution d'une équation aux dérivées partielles (EDP) à la résolution d'un système d'équations linéaires. Les inconnues sont les valeurs de la variable d'état aux noeuds de la discrétisation c'est-à-dire aux sommets des éléments finis. Ensuite, à l'intérieur d'un élément, la variable d'état est approximée en interpolant les valeurs aux noeuds à l'aide d'un polynôme.
Cette méthode numérique a donné naissance à des logiciels de calculs conçus pour traiter de manière générique un grand nombre de problèmes.

Dans un logiciel éléments finis, on distingue traditionnellement 3 modules (Fig. IV.1) [Coulomb81] :

- Le *préprocesseur* : son rôle consiste à définir la géométrie du problème, à décrire les propriétés physiques des différents matériaux et à affecter des conditions aux limites du problème; il fait aussi le choix d'une formulation adaptée. On entend par formulation un modèle simplifié d'EDP. En effet, les équations de Maxwell qui régissent tous les phénomènes électromagnétiques seraient trop complexes à résoudre dans leur intégralité. On effectue donc des hypothèses simplificatrices, suivant le dispositif et les phénomènes étudiés, pour se ramener à une EDP plus simple à traiter.
Enfin le préprocesseur construit le maillage du domaine c'est-à-dire qu'il définit les noeuds et les éléments de la discrétisation. C'est sur cet aspect que nous nous sommes penchés dans ce chapitre.

- Le *processeur central* : il construit puis résout le système linéaire associé au problème. A la fin de cette étape les valeurs aux noeuds sont connues.

- Le *postprocesseur* : celui ci permet d'exploiter au mieux les résultats fournis par le processeur central. Il fournit notamment des outils pour l'analyse du problème : visualisation des lignes de champs, calcul de grandeurs globales comme l'énergie ou la force s'exerçant sur une région volumique, ...

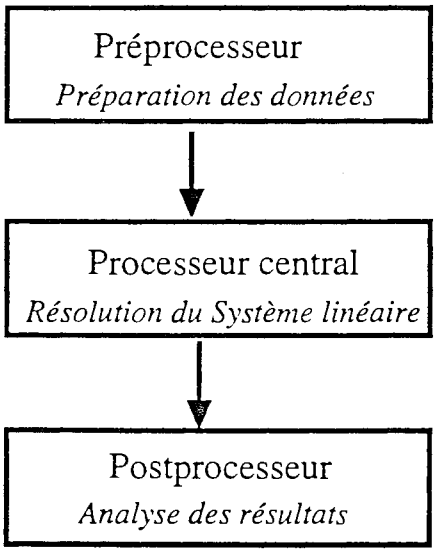


Figure IV.1: Organisation d'un logiciel éléments finis

L'étape de discrétisation du domaine est en fait fondamentale dans la méthode éléments finis car la convergence et la précision des calculs (lors de la résolution du système linéaire) dépendent en grande partie du maillage réalisé par le préprocesseur.
Les principales techniques de générations de maillage peuvent être séparées en 2 familles : les mailleurs structurés [George90], qui nécessitent l'intervention de l'utilisateur et s'appliquent essentiellement à des problèmes possédant une géométrie simple, et les mailleurs automatiques, appelés aussi mailleurs libres, qui limitent considérablement le travail de l'utilisateur. Nous nous intéresserons, ici, uniquement aux maillages constitués d'éléments tétraédriques générés par un mailleur automatique de type Delaunay. Ce type de mailleur, implanté dans le logiciel [FLUX3D] est incontournable en électromagnétisme. Il permet, en effet, de mailler des objets de formes complexes comme l'air entourant un dispositif, problème qui ne se pose pas en mécanique des structures.
Cependant, ce mailleur ne garantit absolument pas un maillage de "bonne qualité" car certains des éléments tétraédriques générés peuvent dégrader la précision des calculs. En effet, il existe de fortes corrélations entre la forme des éléments et le conditionnement du système linéaire traité par le processeur central [Zgainski96]. Une amélioration de la qualité du maillage est donc nécessaire et elle peut être réalisée de deux façons distinctes (Fig. IV.2) :

- La première approche consiste à améliorer le maillage *après résolution*. Dans ce cas, on affine le maillage dans les zones du domaine où la solution est moins précise. Ceci peut être fait en augmentant, dans les mailles incriminées, soit le nombre de noeuds*, soit le degré des polynômes

* Adaptativité h

d'interpolation**. Ce type d'amélioration - à *posteriori* - garantit une solution précise mais d'une part, elle nécessite plusieurs résolutions successives et donc un coût en temps de calcul très important, et d'autre part elle dépend d'un critère d'erreur à adapter à chaque problème physique.

- Dans la seconde approche, l'amélioration est réalisée avant l'étape de résolution. On parle alors d'amélioration a *priori* . Dans cette approche, aucune mesure de l'erreur n'est disponible et il faut par conséquent, pouvoir disposer de critères de qualité d'une maille indépendants du problème traité.

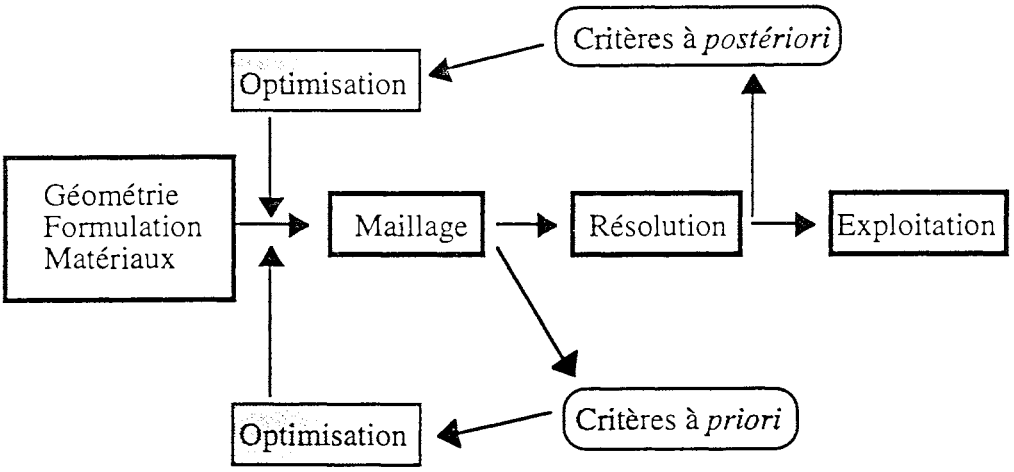


Figure IV.2: Optimisation de maillage a priori et à posteriori

Notre étude portera uniquement sur l'optimisation du maillage a *priori*. En fait, les deux méthodes sont complémentaires mais il est nécessaire de disposer d'un maillage initial de bonne qualité avant de procéder à une optimisation à *posteriori* gourmande en temps de résolution.
Dans une première partie nous présenterons la triangulation de Delaunay utilisée pour générer un premier maillage puis nous décrirons les critères d'optimisation que l'on rencontre dans la littérature.
Dans une seconde partie, nous verrons quelques algorithmes d'optimisation de maillages et enfin nous exposerons un algorithme original basé sur les méthodes génétiques.

** Adaptativité p

2 GENERALITES

2.1 Définitions

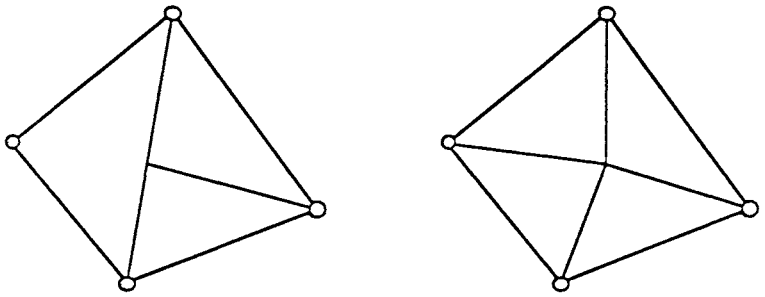
On se donne un domaine polyédral Ω défini par l'ensemble de ses facettes extérieures (segments en dimension 2). Une définition d'un maillage peut être la suivante [George90] :

Définition : On appelle **maillage** M un ensemble d'éléments géométriques E_i tel que :
$\cup E_i = \Omega$
où les E_i sont des tétraèdres , hexaèdres, ... en dimension 3 et des triangles, quadrangles, ... en dimension 2

Définition : Un maillage est dit **conforme** si pour deux éléments quelconques E_i et E_j du maillage l'intersection est [Talon89]:

- soit vide
- soit réduite à un point
- soit réduite à une arête
- soit égale à une facette (en dimension 3 uniquement)

La figure IV.3 illustre la propriété de conformité pour un maillage en dimension 2.



Maillage Non conforme Maillage conforme

Figure IV.3: Conformité d'un maillage en 2D

Dans le cadre de la thèse, nous avons été amenés à travailler sur le logiciel [FLUX3D] développé au Laboratoire d'Électrotechnique de Grenoble. Ce logiciel possède un mailleur structuré et un mailleur automatique.

Le mailleur structuré se base sur des méthodes de transport/projection [George90] ou d'extrusion [Simon93] pour générer des éléments rectangulaires comme des prismes ou des quadrangles. Un avantage de ce type de mailleur est qu'il permet de contrôler facilement le nombre d'éléments générés. De plus, dans les domaines où la variable d'état présente de fortes variations (anisotropie), le mailleur structuré se révèle être mieux adapté [Zgainski96].

Cependant, ce type de mailleur ne peut être utilisé que sur des domaines présentant une géométrie très simple c'est-à-dire là où l'utilisateur peut facilement contrôler la génération des éléments. Par contre

pour des domaines de géométrie complexe (par exemple l'air entourant le dispositif), un mailleur automatique est indispensable.
Dans FLUX3D, le mailleur automatique se base sur la triangulation de Delaunay que nous allons exposer brièvement. Cette méthode génère des triangles en dimension 2 et des tétraèdres en dimension 3.

2.2 Maillage de Delaunay

La triangulation de Delaunay [Hermeline82] est très générale et peut s'appliquer à des espaces de dimension quelconque. Pour simplifier, les définitions qui suivent seront exposées pour la dimension 2 ou la dimension 3 et illustrées par des figures en dimension 2.

Définition : Enveloppe convexe (Fig IV.4)

On appelle **enveloppe convexe** de N noeuds, le plus petit volume V (en 3D) ou la plus petite surface S (en 2D) qui s'appuie sur un sous ensemble de ces N noeuds et tel que toutes les arêtes joignant 2 noeuds de ce sous ensemble soient incluses dans V (en 3D) ou S (en 2D).

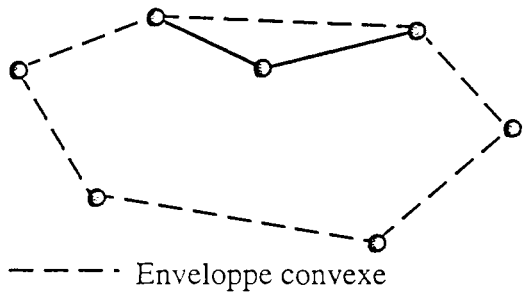


Figure IV.4: Enveloppe convexe d'un ensemble de 7 noeuds

Définition : Maillage de Delaunay (Fig. IV.5)

Soit un ensemble de N noeuds d'un domaine V. Le maillage de l'enveloppe convexe de V, est dit **maillage de Delaunay** si pour tous les éléments E_i la boule circonscrite à E_i est vide. Autrement dit, la boule circonscrite à E_i ne doit contenir aucun autres noeuds que ceux de E_i.

En dimension 3, la boule circonscrite à l'élément tétraédrique est une sphère. En dimension 2, la boule circonscrite à l'élément triangulaire est un cercle.

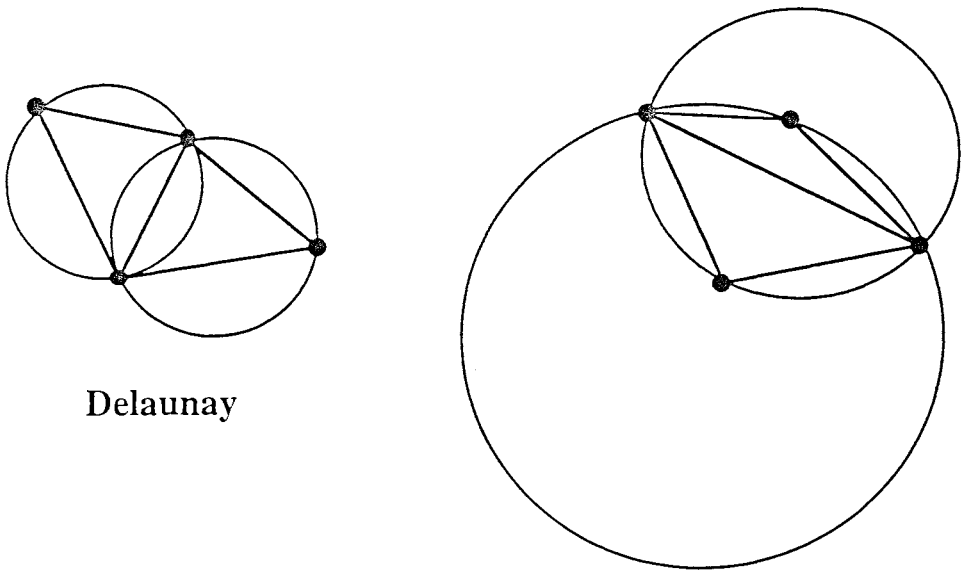


Figure IV.5: Deux exemples de triangulation

Pour construire une triangulation de Delaunay, on procède en général de façon itérative [Hermeline82]. A partir d'une triangulation de Delaunay d'un ensemble de N noeuds, on insère le (N+1) i^{eme} noeud de la discrétisation à l'aide d'un remaillage local. En fait, lors de l'insertion du (N+1) i^{eme} noeud, trois cas peuvent se présenter (Fig. IV.6) :

- **Cas (a)** : Le noeud est intérieur à certains éléments (et donc intérieur à leurs sphères circonscrites).
- **Cas (b)** : Le noeud n'appartient à aucune sphère.
- **Cas (c)** : Le noeud est intérieur à certaines sphères mais n'appartient à aucun élément.

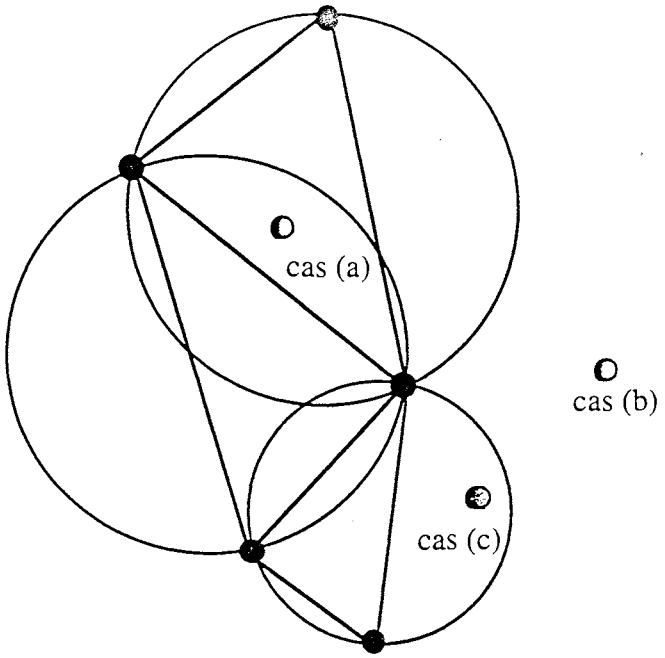


Figure IV.6: Insertion d'un nouveau noeud dans une triangulation de Delaunay

Pour les cas (**a**) et (**c**) , le traitement est sensiblement identique (Fig. IV.7 et IV.9) : il consiste à détruire les éléments dont la sphère circonscrite contient le noeud et à générer de nouveaux éléments qui s'appuient sur ce noeud et sur l'ensemble des facettes (arête en 2D) "vues" par le noeud.
Dans la situation (**b**) (Fig. IV.8) aucun élément de la triangulation précédente n'est détruit.

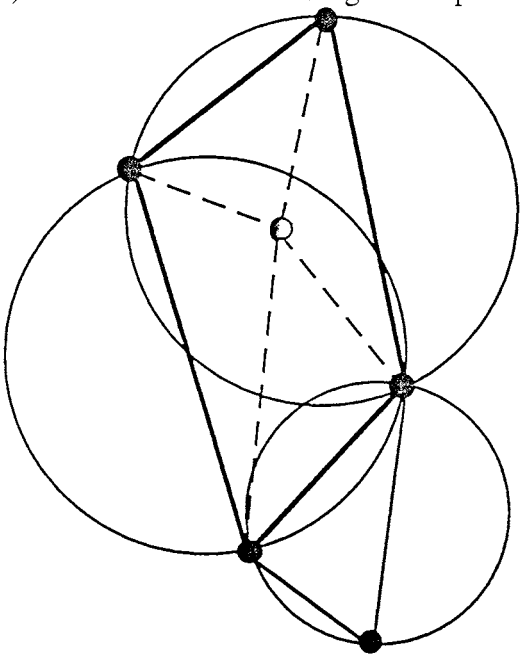


Figure IV.7: traitement du cas (a)

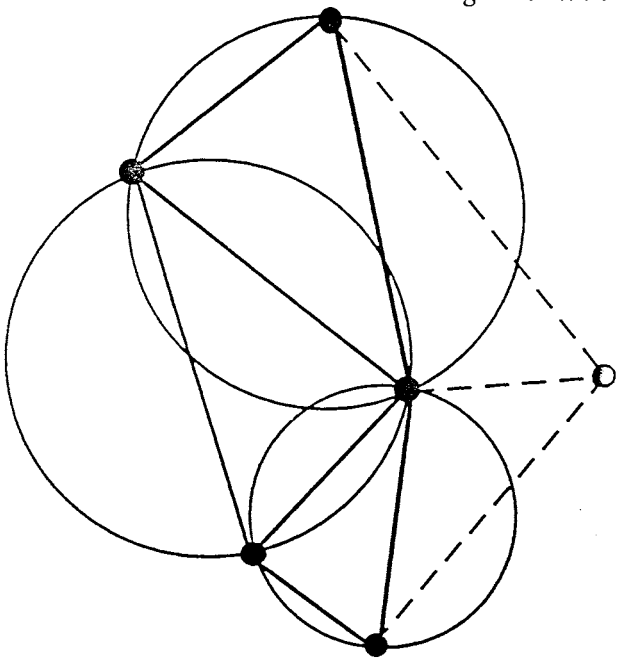


Figure IV.8: traitement du cas (b)

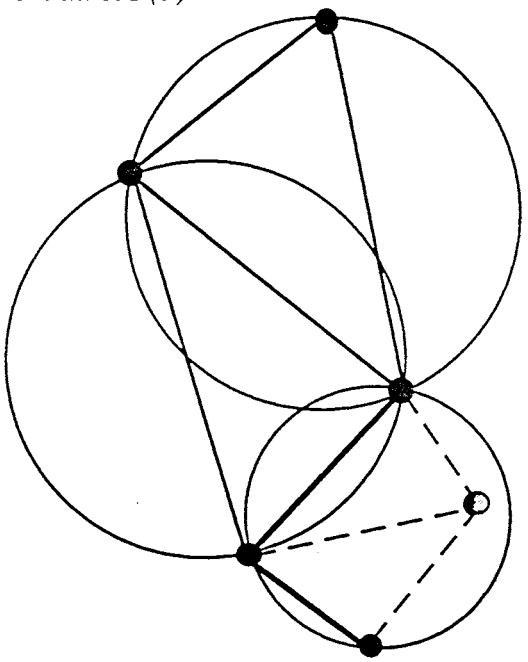


Figure IV.9: traitement du cas (c)

Dans le logiciel FLUX3D, la géométrie est décrite par un modèle B.Rep* (ou Représentation Frontière) : l'utilisateur définit les points puis les lignes de la géométrie de son dispositif. Les faces et les volumes sont ensuite construits automatiquement par "topologie". Cette représentation *hiérarchique* de la géométrie est particulièrement bien adaptée à un mailleur libre de type Delaunay.

* Boundary Representation

En effet, FLUX3D va mailler de manière automatique et chronologique, les lignes, les faces puis les volumes. En pratique, pour mailler une face du domaine, FLUX3D construit une surface convexe englobant la face et, en s'appuyant sur les noeuds de la discrétisation des lignes, il génère une triangulation de Delaunay. Pour traiter des faces non convexes, le mailleur rajoute des noeuds sur les lignes dans le but d'assurer la conformité du maillage de la face. Une fois les faces traitées, FLUX3D génère les éléments tétraédriques dans les volumes de la même façon, mais en s'appuyant cette fois ci sur les noeuds des faces.
L'avantage de la triangulation de Delaunay est qu'elle permet de générer de façon automatique un maillage dans des volumes de géométrie complexe. Par contre, l'utilisateur ne peut pas contrôler facilement la densité des noeuds dans le maillage et encore moins la forme des éléments générés. La figure IV.10 présente une vue partielle du maillage tétraédrique d'une machine synchrone à aimants.

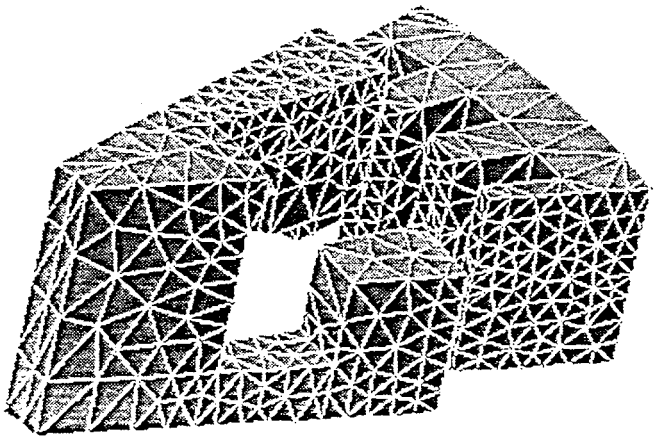


Figure IV.10: Maillage tétraédrique d'une machine synchrone à aimants

3 AMELIORATION DE MAILLAGES A *PRIORI*

3.1 Introduction

Dans la méthode éléments finis, la précision des calculs effectués par le processeur central dépend en grande partie du maillage réalisé par le préprocesseur. Le maillage M doit satisfaire certains critères :

- M doit être conforme.
- M doit être dense là où la variable d'état présente de grandes variations.
- M doit posséder le moins possible d'éléments de "mauvaise qualité".

Le premier critère est contrôlé par FLUX3D lors de la génération automatique basée sur la

triangulation de Delaunay. Le second critère suppose une bonne connaissance physique du problème. Il peut être plus ou moins contrôlé par l'utilisateur qui, lors de la génération de maillage, peut définir le nombre de noeuds qu'il désire sur les lignes de sa géometrie. Quant au troisième critère, un **élément est** considéré comme étant de mauvaise qualité ou **impropre au calcul élément finis si sa forme géométrique s'éloigne trop de l'équilatéralité**. En effet, il existe de fortes corrélations entre le conditionnement numérique du système linéaire (et donc la précision des résultats) et la forme géométrique des éléments du maillage [Zgainski96]: plus les éléments sont proches de l'équilatéralité, meilleur devient le conditionnement. Malheureusement, par soucis d'efficacité, la forme des éléments n'est pas contrôlée lors de la génération automatique du maillage. La figure IV.11 montre la distribution des mailles obtenues lors de la discrétisation d'un problème constitué par un moteur synchrone à aimants. Il apparaît clairement que des mailles en nombre important ont des formes très éloignées de l'équilatéralité.

Il est donc souhaitable d'effectuer un certain nombre de corrections sur le maillage avant de résoudre le système linéaire. Pour cela on met en œuvre des algorithmes d'amélioration qui utilisent des critères de qualité qui rendent compte de l'équilatéralité des éléments. Nous allons maintenant présenter les divers critères que l'on rencontre dans la littérature.

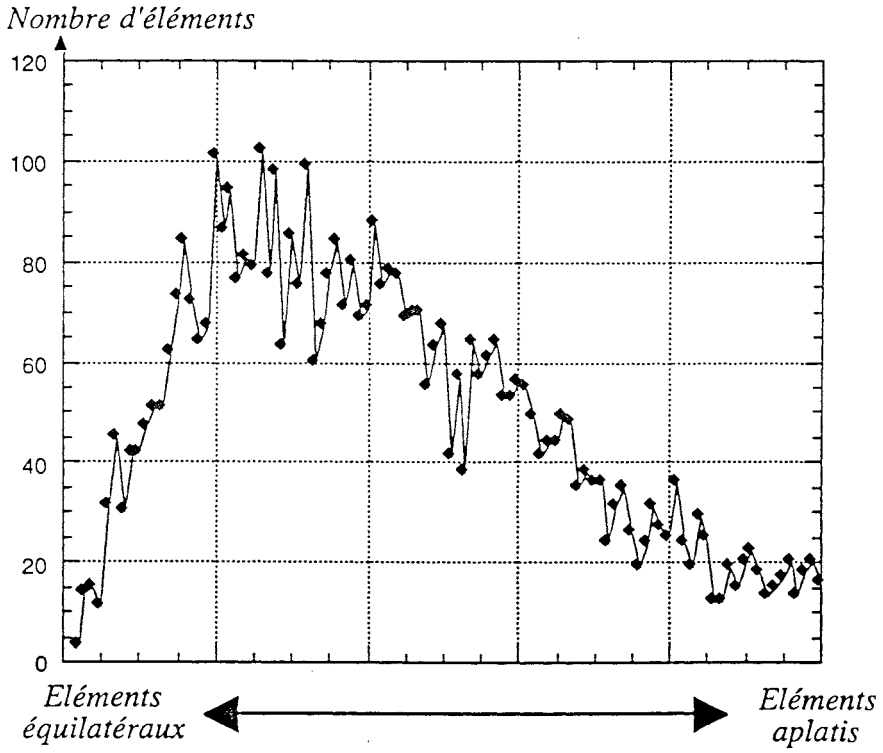


Figure IV.11: Distribution de la forme des éléments

3.2 Qualité d'un élément tétraédrique

Les critères C que nous allons voir, traduisent la forme géométrique de l'élément tétraédrique. Un élément sera de grande qualité si C est proche de la valeur du critère pour l'élément équilatéral noté C^*. Dans [Rassineux95] et [Henot93] on trouve le critère:

$$C = \frac{\rho}{h}$$

où ρ est le rayon de la sphère inscrite au tétraèdre et h la longueur de son plus grand coté.
On peut alors définir la qualité Q de l'élément par $Q = C / C^*$.
Dans [FLUX3D] ou [Talon89] le critère proposé est le suivant :

$$C = \frac{Volume\ Sphère\ Inscrite}{Volume\ du\ Tétraèdre}$$

Pour le tétraèdre équilatéral on a: $C^* = \frac{\pi}{6\sqrt{3}}$

La qualité s'écrit donc:

$$Q = \frac{6\sqrt{3}}{\pi} \frac{Volume\ Sphère\ Inscrite}{Volume\ du\ Tétraèdre}$$

Si on pose V le volume du tétraèdre et S sa surface alors ce même critère peut s'écrire :

$$C = \frac{36\pi V^2}{S^3}$$

et la qualité

$$Q = \frac{216\sqrt{3}\,V^2}{S^3}$$

On trouve aussi le critère [Cavendish85] :

$$C = \frac{Rayon\ \ Sphère\ Inscrite}{Rayon\ \ Sphère\ Circonscrite}$$

On peut se demander si ces critères sont effectivement tous équivalents, c'est-à-dire, s'ils traduisent de la même manière un changement dans la forme de l'élément. [Parthasarathy93] a étudié la sensibilité de ces critères en soumettant un élément tétraédrique à certains tests de distorsion (Fig. IV.12).

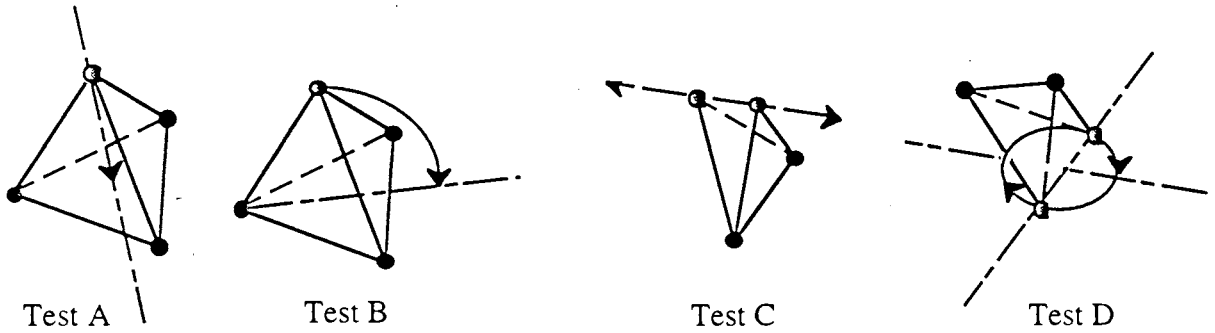
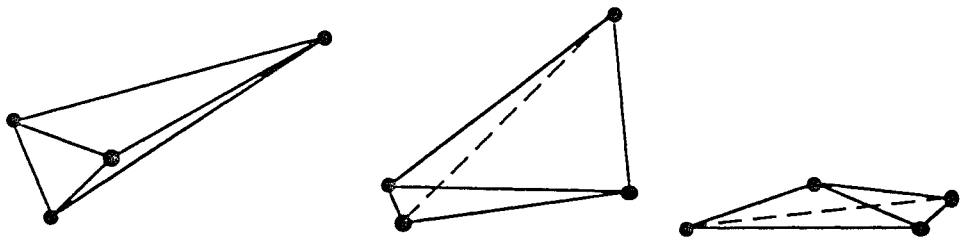


Figure IV.12: Tests de déformation sur l'élément tétraédrique

Ces tests rendent compte des déformations que subissent les tétraèdres lors d'une génération automatique du maillage. Ils permettent, entre autres, de comparer la sensibilité des critères lorsqu'un élément tétraédrique équilatéral dégénère en un des éléments de mauvaise qualité que l'on rencontre

communément (Fig. IV.13).



élément "*fin*" élément "*allongé*" élément "*aplati*"

Figure IV.13: Quelques éléments de mauvaises qualité

Les critères étudiés par [Parthasarathy93] sont présentés dans la Table IV.1.

Critères	Valeur du critère pour l'élément équilatéral	Références
$\alpha = \frac{(Moyenne\ de\ la\ Longueur des\ arêtes)^3}{Volume\ Tétraèdre}$	$\alpha^* = 8.479670$	[Dannelongue91]
$\beta = \frac{Rayon\ \ Sphère\ Inscrite}{Rayon\ Sphère\ Circonscrite}$	$\beta^* = 3.0$	[Cavendish85]
$\omega = \frac{Rayon\ de\ sphère\ circonscrite}{Longueur\ du\ plus\ grand\ côté}$	$\omega^* = 0.612507$	[Baker89]
$\tau = \frac{Longueur\ du\ plus\ grand\ côté}{Longueur\ du\ plus\ petit\ côté}$	$\tau^* = 1.0$	[Baker89]
$\kappa = \frac{(Volume\ Tétraèdre)^4}{\left(\sum_1^4 (Surface\ des\ facettes)^2\right)^3}$	$\kappa^* = 4.58457\ 10^{-4}$	[Graichen91]
$\gamma = \frac{\sqrt[3]{\frac{1}{6}\sum_1^6 (Longueur\ des\ arêtes)^2}}{Volume\ Tétraèdre}$	$\gamma^* = 8.479670$	[Cougny90]
$\sigma = \frac{Longueur\ du\ plus\ grand\ coté}{Rayon\ de\ la\ Sphère\ Inscrite}$	$\sigma^* = 4.898979$	[Henot93]

Table IV.1: Critères géométriques pour l'optimisation de maillage

Les résultats qu'il a obtenu sont montrés sur les figures IV.14 et IV.15. On constate, que seuls les critères α, β, σ et γ sont capables de caractériser correctement les 4 tests de déformation. Nous

avons réalisé ces mêmes tests pour le critère de Talon implanté dans le logiciel [FLUX3D]. Les courbes obtenues ont été comparées avec celles du critère de Henot (σ) et elles sont présentées sur les figures IV.16 et IV.17. On peut voir que les comportements de ces deux critères sont sensiblement les mêmes: on peut donc inclure le critère de FLUX3D dans la liste des "bons" critères de Parthasarathy.

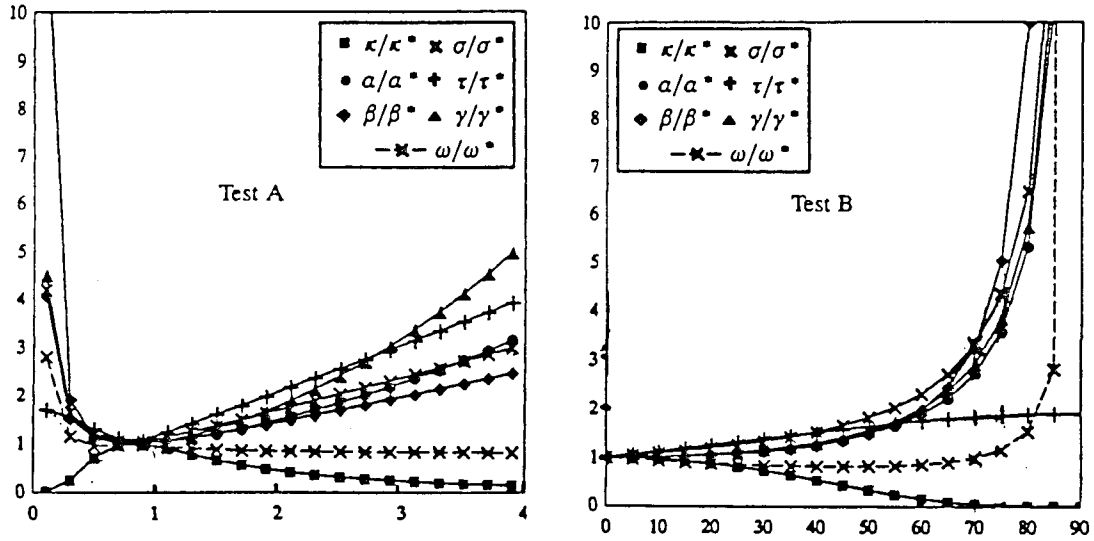


Figure IV.14: Evolution des critères de la table IV.1 pour les tests A et B

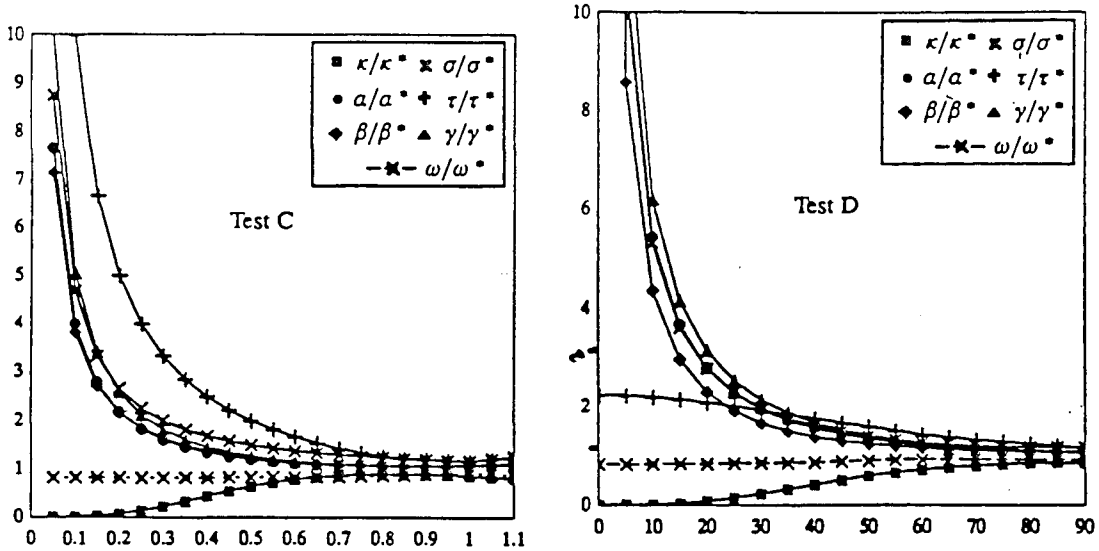


Figure IV.15: Evolution des critères de la table IV.1 pour les tests C, D

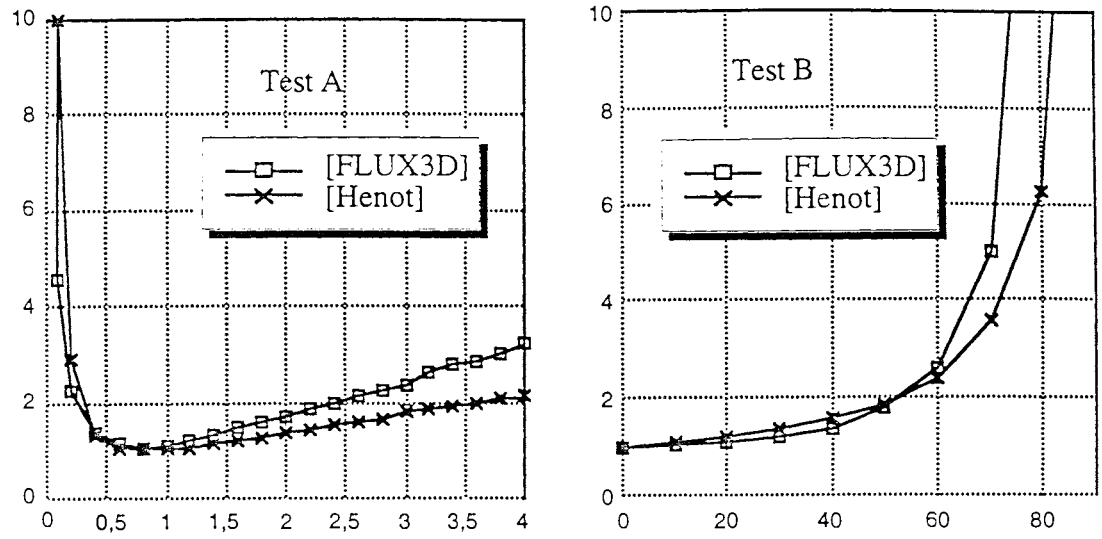


Figure IV.16: Evolution des critères de FLUX3D et de Henot sur les tests A et B

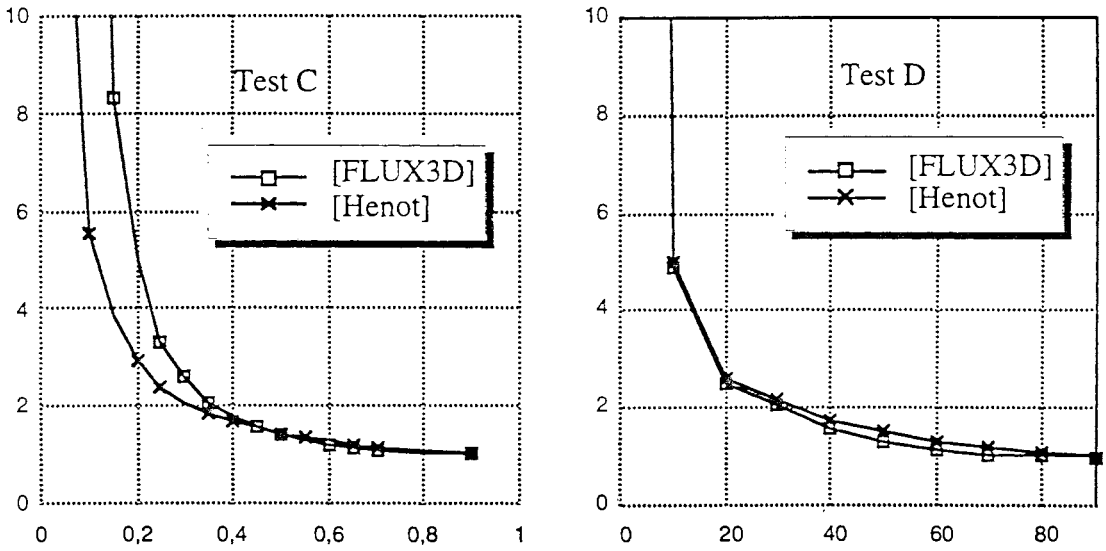


Figure IV.17: Evolution des critères de FLUX3D et de Henot sur les tests C et D

3.3 Algorithmes d'amélioration de maillage

Nous venons de voir différents critères qui permettent de définir la qualité d'un élément. Il nous faut maintenant définir une fonction objectif dans le but d'optimiser le maillage. Pour cela, nous prendrons par exemple la fonction:

$$f = \frac{1}{N}\sum_{i=1}^{i=N}(1-Q_i)$$

où Q_i est la qualité de l'élément i dans le maillage et N le nombre total d'éléments. Q_i peut prendre une des expressions vues précédemment.

Un algorithme d'optimisation essaiera donc de minimiser f c'est-à-dire la qualité moyenne des éléments. La valeur idéale de f (0 !!) correspond à une configuration du maillage dans laquelle tous les éléments sont équilatéraux. On peut aussi utiliser des fonctions qui n'accordent pas le même poids aux différents éléments. Par exemple, pondérer plus fortement les éléments dont la qualité dépasse une certaine valeur seuil. Cependant, dans notre étude, nous nous sommes limités à une fonction attribuant des poids identiques à toutes les mailles.

Les algorithmes d'optimisation, que nous allons voir, mettent en jeu des outils de modification locale du maillage. Ces outils peuvent toutefois être divisés en deux grandes familles: les outils de modification *topologique* qui jouent sur les arêtes ou les facettes du maillage et les outils de modification *non topologique* qui jouent sur les coordonnées des noeuds sans toucher à la topologie du maillage.

3.3.1 Outils de modification topologique

[Talon89] a proposé deux opérations élémentaires sur les arêtes. La première est l'opération **2T->3T,** qui détruit 2 tétraèdres partageant une face, en insérant une arête pour former 3 nouveaux tétraèdres (Fig. IV.18).

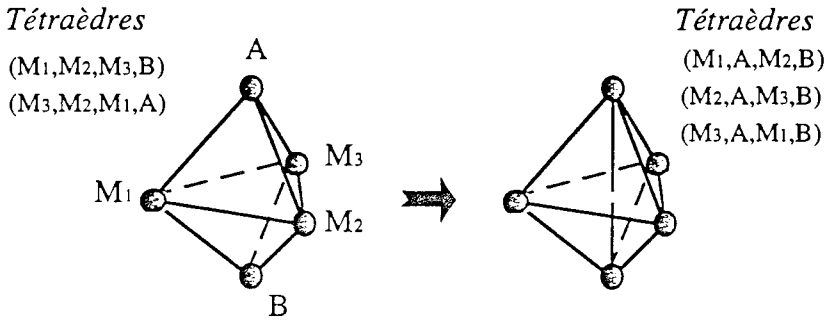


Figure IV.18: Opération 2T->3T

Cette opération n'est possible que si l'arête [AB] coupe effectivement la face commune (M_1,M_2,M_3).
L'autre opération (**3T->2T**) est en fait la réciproque de la première : elle consiste à supprimer une arête autour de laquelle s'appuient 3 tétraèdres. Cette opération génère alors 2 nouveaux tétraèdres.
Dans [Henot93] on trouve une généralisation de l'opération de suppression d'arête : si [AB] est une arête libre* et si C est la coquille de tétraèdres s'enroulant autour de cette arête alors supprimer [AB] revient à proposer toutes les tétraédrisations qui ne font pas apparaître [AB]. La figure IV.19 présente la suppression d'une arête dans une coquille de 4 tétraèdres et les 2 possibilités de remaillage ne faisant plus apparaître l'arête.

* non frontalière

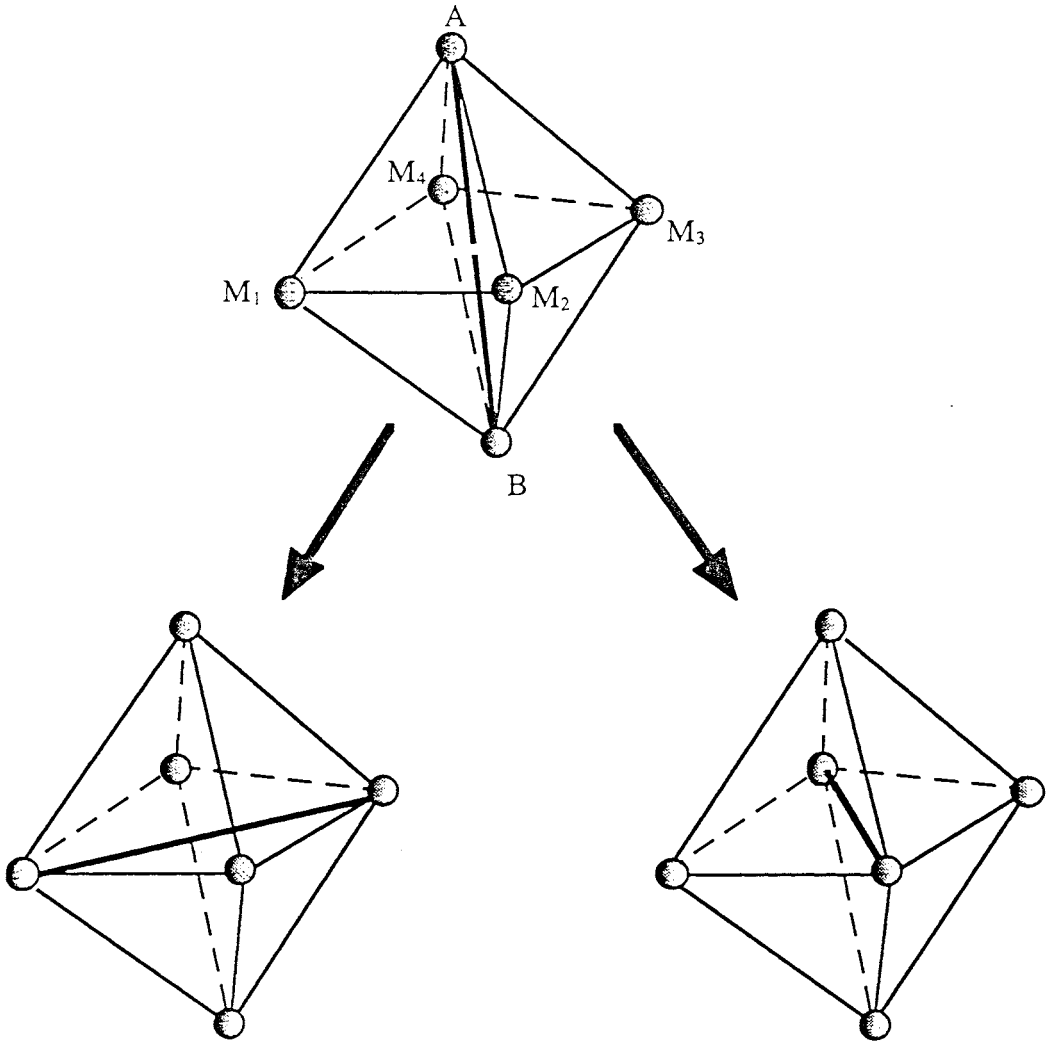


Figure IV.19: Tétraédrisations dans une coquille de quatre éléments

Dans le cas d'une coquille *convexe* $(M_i,A,M_{i+1},B)_{i=1,n}$ de n tétraèdres, le nombre de remaillages possibles N_n se calcule par la formule récurrente :

$$N_n = \sum_{i=3}^{i=n} N_{i-1} N_{n+2-i} \text{ avec } N_2 = 1 \text{ [Henot93]}$$

En fait, remailler une coquille de tétraèdres revient à "retrianguler" le polygone formé par les noeuds (M_i). La figure IV.20 illustre les remaillages possibles d'une coquille de 5 tétraèdres.

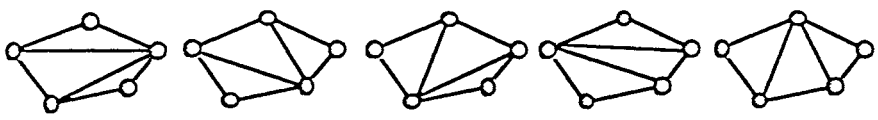


Figure IV.20: Triangulations possibles d'un polygone de 5 noeuds

Soit NT_n le nombre de triangles différents constructibles dans un polygone de n noeuds, alors on a la formule : $NT_n = \frac{n(n-1)(n-2)}{6}$. On s'aperçoit (Table IV.2) que NT_n croît beaucoup moins vite que N_n. Cela permet d'éviter des re-calculs (coûteux !) du critère de qualité notamment lorsqu'on recherche une configuration optimale de la coquille. On peut aussi noter que l'opération **3T->2T** de Talon n'est qu'un cas particulier de la suppression d'arête où n vaut 3.

n	3	4	5	6	7	8	9	10
N_n	1	2	5	14	42	132	429	1430
NT_n	1	4	10	20	35	56	84	120

Table IV.2: Configurations possibles dans une coquille de tétraèdres

Dans le cas où la coquille est *non convexe* certains triangles sont interdits dans la retriangulation : on ne fait donc que restreindre le nombre de remaillages possibles.
Dans les tables IV.3 et IV.3bis nous présentons quelques statistiques sur le nombre et la taille des coquilles de tétraèdres s'enroulant autour des arêtes libres d'un maillage. Ces résultats ont été obtenus pour l'air entourant un dispositif dans 2 problèmes tests différents (voir par ailleurs).

Taille de la coquille	3	4	5	6	7	8	9	10	≥ 11	Total
Nombre de coquilles	273	505	489	403	209	62	29	8	2	1980
%	14%	26%	25%	20%	11%	3%	1%	0.4	0.1	100%

Table IV.3: Statistiques sur le nombre de coquilles convexes

Taille de la coquille	3	4	5	6	7	8	9	10	≥ 11	Total
Nombre de coquilles	519	979	1012	868	453	134	44	13	5	4027
%	13%	24%	25%	22%	11%	3%	1%	0.3	0.1	100%

Table IV.3bis: Statistiques sur le nombre de coquilles convexes

On peut encore citer des outils d'insertion ou de suppression de noeuds dans le maillage. La suppression peut être faite lorsqu'un noeud est entouré par exactement 4 tétraèdres (Fig. IV.21). Quant à l'insertion, elle peut avoir lieu sur une arête jugée trop longue par rapport aux autres arêtes de la coquille.

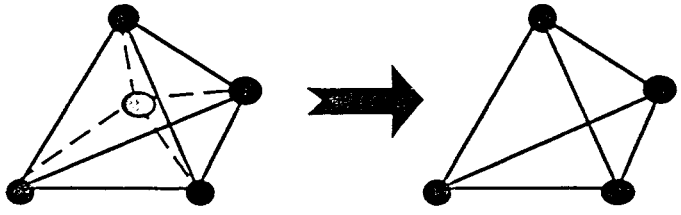


Figure IV.21: Suppression d'un noeud

3.3.2 Algorithmes d'optimisation s'appuyant sur ces outils

Nous disposons maintenant d'une fonction objectif et d'outils autorisant des modifications topologiques du maillage. Cette section décrit quelques algorithmes généraux d'amélioration du maillage. [Talon89] a proposé un algorithme descendant dit **Glouton** qui parcourt tous les tétraèdres T_i du maillage en leur appliquant une des opérations 2T<->3T. Les nouveaux tétraèdres sont acceptés si et seulement si la qualité moyenne du maillage s'améliore. Finalement l'algorithme se termine lorsqu'il n'existe plus aucune possibilité de modification topologique améliorant l'objectif.

[Henot93] a proposé un algorithme descendant un peu plus complet qui prend la forme suivante :

Tant que des améliorations sensibles du maillage se produisent faire:

- Pour toutes les arêtes libres du maillage, supprimer l'arête si une nouvelle configuration des tétraèdres s'enroulant autour d'elle améliore la qualité de la coquille.
- Pour toutes les faces non frontalières du maillage, effectuer l'opération 2T->3T si la qualité des 3 nouveaux tétraèdres améliore localement l'objectif.
- Pour tous les noeuds non frontaliers sur lesquels s'appuient 4 tétraèdres, supprimer le noeud si le tétraèdre généré améliore la qualité.
- Pour toutes les arêtes libres, insérer des noeuds si le critère s'améliore.

Pour terminer, on peut citer l'algorithme basé sur le **Recuit Simulé** [Talon89]. Comme nous l'avons vu au chapitre I, cette méthode est très générale et demande, pour chaque problème, que soient précisés un certain nombre de paramètres. Dans l'algorithme de Recuit de Talon, l'**opération de voisinage**, qui permet de perturber la solution courante, est une des modifications réciproques d'ajout et de suppression d'arêtes. Ainsi, un tétraèdre T_i est choisi aléatoirement et on lui applique une des opérations 2T<->3T, chacune d'elles ayant la même probabilité d'être sélectionnée.

- si l'opération d'**ajout** est sélectionnée, on choisit un voisin de T_i partageant une face commune et qui autorise l'insertion de l'arête. Si aucun des tétraèdres voisins ne permet de conserver la réalisabilité du maillage, c'est-à-dire si l'arête à insérer ne coupe pas la face commune, alors on abandonne T_i et on choisit un autre tétraèdre.

- si l'opération de **suppression** est sélectionnée, on choisit 2 tétraèdres voisins de T_i qui finissent d'entourer une des arêtes de T_i. On peut alors supprimer l'arête. Il se peut qu'aucune des arêtes de T_i ne soit complètement entourée, alors, comme dans le cas précédent, on sélectionne un nouveau tétraèdre.

La **température initiale** doit être prise suffisamment élevée pour que toutes les solutions réalisables,

même celles dégradant la fonction objectif, aient une probabilité proche de 1 d'être atteinte. [Talon89] propose de choisir la température initiale T_0 de telle sorte que :

$$exp\left(\frac{-3}{(nT_0)}\right) = P_{deb}$$

où n désigne le nombre de tétraèdres dans le maillage initial et P_{deb} est un paramètre choisi égal à 0,8. Cette équation signifie qu'en début d'exécution on accepte, avec une probabilité P_{deb}, des solutions réalisables qui dégradent l'objectif de 3 unités.

La **loi de décroissance** de la température est géométrique : $T^{k+1} = \alpha T^k$ avec $\alpha < 1$. Dans [Talon89] la température est abaissée à chaque fois qu'une transformation sur le tétraèdre est possible. Mais cela ne veut pas dire que la longueur de la chaîne de Markov est égale à un : en effet, les deux outils de modification 2T<->3T peuvent générer des configurations non réalisables.

Enfin les **critères d'arrêt** de l'algorithme sont de deux types :
- soit la température du système atteint une température finale T_f.
- soit le nombre d'itérations sans changement dans la topologie du maillage atteint une valeur N.

[Talon89] détermine T_f de sorte que :

$$exp\left(\frac{-0.01}{(n_c T_f)}\right) = P_{fin}$$

où n_c désigne le nombre de tétraèdres dans le maillage courant et P_{fin} une probabilité de valeur faible (0,1 par exemple). A la température finale, on s'assure donc que seules des dégradations très faibles de l'objectif, de l'ordre de 0,01, sont acceptées avec une probabilité P_{fin} elle même très faible.
Pour le second test, N est choisi proportionnel au nombre d'éléments dans le maillage initial :

$$N = K_N n$$

K_N est un paramètre de l'algorithme qui pourra dépendre du domaine étudié.

Les algorithmes de type descendant ont donné des résultats très convaincants [Henot93] [Talon89]: la qualité moyenne du maillage s'améliore en même temps que les éléments de très mauvaise qualité disparaissent. En général, ces algorithmes convergent vers une topologie stable, pour laquelle plus aucune modification ne provoque d'amélioration, en 3 ou 4 itérations.
Par contre, l'algorithme de Recuit Simulé n'apporte pas de réelles améliorations par rapport à l'algorithme glouton. Cet algorithme est de plus très gourmand en temps de calcul (près de 30 fois plus de temps que l'algorithme glouton pour une même qualité moyenne) et son emploi ne se justifie donc pas [Talon89].
Ces algorithmes ont été présentés car, selon leurs auteurs, les algorithmes d'amélioration non topologique, que nous allons voir, ne deviennent intéressants que lorsqu'ils sont appliqués à la suite d'une optimisation topologique. Pour cette raison, nous avons implanté un algorithme descendant de

modification topologique. Il s'agit de l'algorithme de [Henot93] dans lequel la suppression d'arête n'est réalisée que pour des coquilles de 3 tétraèdres (c'est-à-dire l'opération 3T->2T de Talon). Cet algorithme est implanté pour pouvoir comparer dans de "bonnes conditions" les algorithmes de modification non topologique traditionnels avec notre algorithme génétique.

3.3.3 Algorithmes d'amélioration non topologique

Contrairement aux outils précédents, ceux-ci ne touchent pas à la topologie du maillage : ils ne font que déplacer les noeuds sans modifier leur connectivité. Nous présentons l'algorithme de barycentrage ainsi que l'algorithme de bougé de point qui semble être plus performant.

A Barycentrage

Le barycentrage [Albertini88] consiste à déplacer un noeud P vers le barycentre de ses noeuds voisins (c'est-à-dire les noeuds connectés à P par une arête)(Fig. IV.22 en 2D)

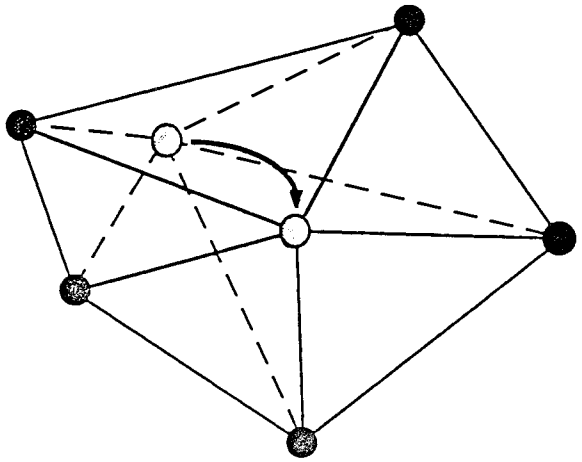


Figure IV.22: Barycentrage

Un algorithme d'optimisation consiste donc à réaliser le barycentrage sur tous les noeuds non frontaliers du maillage. [Talon89] constate que cet algorithme est performant en 2D mais que dans le cas 3D il peut entraîner une dégradation de l'objectif.

B Bougé de points

Dans le bougé de points de [Henot93], on considère un noeud non frontalier et la boule B de tétraèdres s'appuyant sur ce noeud. On déplace alors le noeud P vers un point optimal P_{opt} calculé à partir des tétraèdres idéaux s'appuyant sur les facettes extérieures de B. Schématiquement :

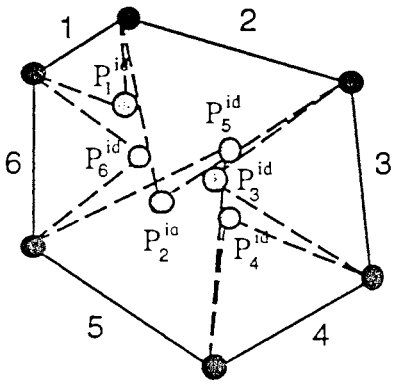
$$P = P + a\overrightarrow{PP_{opt}}$$

avec P_{opt} isobarycentre des points P_j^{id} où P_j^{id} est la position idéale de P pour la facette numéro *j* de B:

en fait P_j^{id} est une position de P qui donnerait un tétraèdre s'appuyant sur la facette *j* de bonne qualité. Pour cela on peut choisir pour P_j^{id} un point situé sur la normale partant du centre de gravité de la facette j (Fig. IV.23 en 2D) [Rassineux95].
Le coefficient "a" représente un pas de déplacement dans la direction de P_{opt} . La valeur de "a" peut être déterminée par une recherche dichotomique (Fig. IV.24):

initialement a varie dans l'intervalle [bmin,bmax] avec bmin = 0 et bmax = 1
Tant que la qualité de la coquille s'améliore de façon significative **faire**
a = 1/2(bmin+bmax)
On **évalue** la qualité de la coquille pour la position de P = P + aPP_{opt} : **Q(a)** et la qualité pour une valeur du pas a-ε : **Q(a-ε)**.
Si Q(a-ε) < Q(a) **alors** bmin = a
sinon bmax = a
Fintq

Figure IV.24: Calcul du pas de déplacement optimal par dichotomie



O Points idéaux

Figure IV.23 : Construction des points idéaux

Ces deux algorithmes ont été implantés dans le logiciel FLUX3D. Nous avons étudié leur comportement sur deux problèmes tests. On s'aperçoit qu'utilisés seuls (Fig. IV.25 et IV.26), ces algorithmes ne permettent pas d'amélioration sensible du maillage. L'algorithme de barycentrage dégrade même cette qualité sur les deux problèmes. Ceci confirme les résultats de [Talon89] et nous amène à associer systématiquement ces algorithmes de modification non topologique à un algorithme de modification topologique.

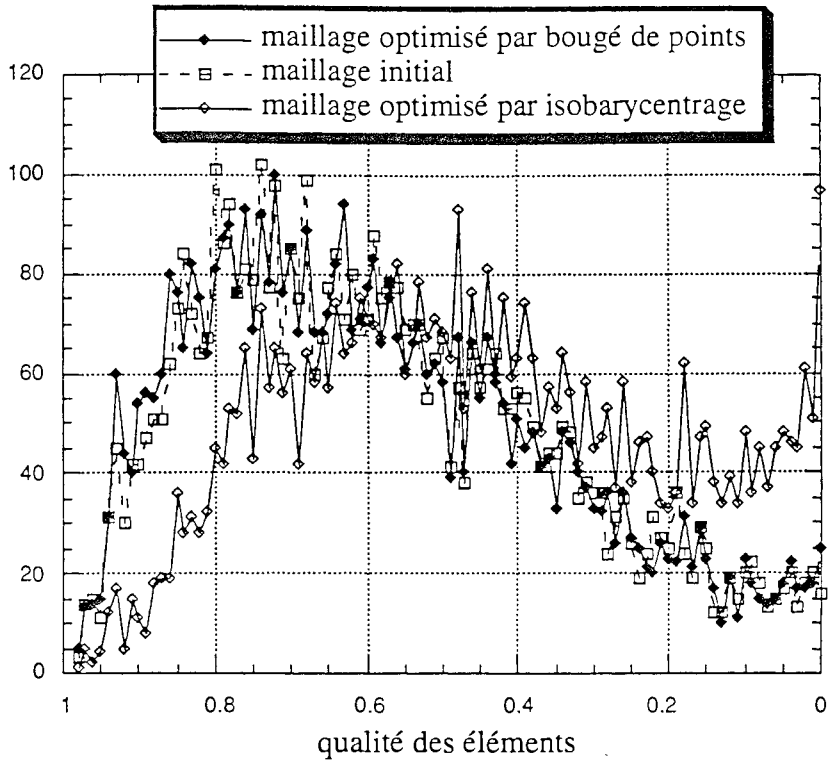


Figure IV. 25: Optimisation non topologique sur le 1er problème

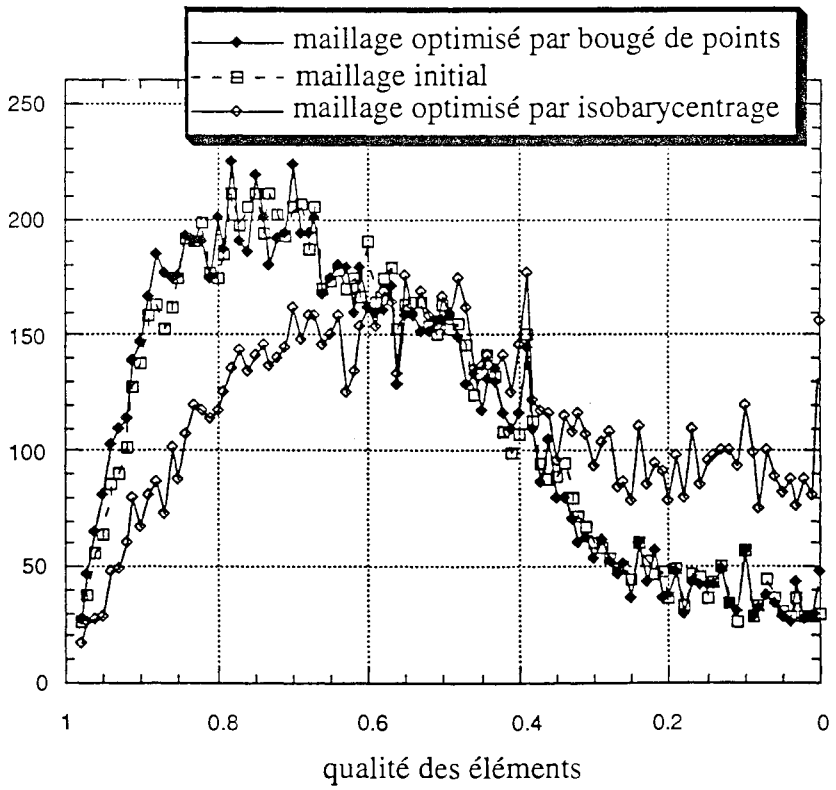


Figure IV.26: Optimisation non topologique sur le 2eme problème

L'autre point important réside dans le traitement séquentiel des noeuds. Ce traitement confère à ces algorithmes non topologique un caractère local pouvant déboucher sur une configuration non-optimale des noeuds dans le maillage. En effet, lorsque les noeuds sont bougés de façon séquentielle, le déplacement de chaque nouveau noeud modifie la position des noeuds précédents dans le maillage. Par conséquent, ces derniers ne se trouvent plus dans leur position "idéale" ou à l'isobarycentre de leurs noeuds voisins, ce qui peut entraîner l'apparition de singularités (tétraèdres aplatis...). Il faudrait un algorithme qui traite de manière globale les positions de tous les noeuds dans le maillage. En 2D, il suffit d'imaginer le domaine à mailler constitué de trous correspondant aux positions des noeuds pour lesquels le maillage est de plus grande qualité. La distribution de ces trous nous est bien sûr inconnue. A partir de là, l'utilisateur possède des billes - les noeuds - qu'il veut lancer sur le domaine en espérant qu'elles iront se placer dans ces trous. Cette opération peut être répétée un certain nombre de fois et, en conservant la qualité des maillages ainsi générés, on pourra orienter les nouveaux lancés de façon plus efficace. Ce type de processus n'est en fait rien d'autre qu'un algorithme génétique.

C Algorithme génétique pour l'optimisation de maillages

Dans notre étude, on s'intéressera uniquement aux noeuds intérieurs aux volumes. Les noeuds situés aux interfaces de volumes seront considérés comme fixes. Un individu sera représenté par une chaîne dans laquelle seront concaténés les vecteurs déplacements de chacun des noeuds par rapport à la position des noeuds dans le maillage initial. Ce sont ces vecteurs déplacements qui seront amenés à évoluer lors de l'exécution de l'algorithme génétique. Ainsi, chaque **individu représente une disposition particulière des noeuds** dans le maillage. L'évaluation d'un individu consiste alors, à récupérer les vecteurs déplacements de chaque noeud, à les ajouter aux coordonnées des noeuds dans le maillage initial et à calculer la qualité moyenne du maillage en utilisant un des critères décrits plus haut.

La principale difficulté réside dans le choix de contraintes sur les vecteurs déplacements de chaque noeud. En effet, des déplacements trop importants de plusieurs noeuds peuvent entraîner le retournement de certains tétraèdres. Mathématiquement cela se traduit par des tétraèdres de volume négatif.

Peut on utiliser alors un codage binaire et les opérateurs classiques de Holland ? Cela revient à coder chacune des composantes des vecteurs déplacements à l'aide d'un certain nombre de bits (Fig. IV.27).

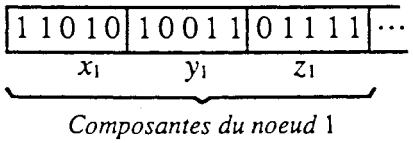


Figure IV.27: Codage binaire des vecteurs déplacements

Un crossover, qui viendrait couper n'importe où dans la chaîne, ou, une mutation agissant sur les bits de poids forts, générerait alors, dans la plupart des cas, des individus dont les tétraèdres seraient

retournés. En effet, il n'est pas possible de contrôler, lors de ces opérations, une contrainte du type :

$$x_i^2 + y_i^2 + z_i^2 \leq f(noeuds\ voisins)\ \forall i$$

où (x_i, y_i, z_i) représente le vecteur déplacement du noeud i et f une fonction qui dépend de la position des noeuds voisins du noeud i et qui majore le déplacement de ce dernier pour éviter un éventuel retournement de tétraèdre.

Pour contourner ces problèmes, on peut bien sûr pénaliser fortement ces individus mais, comme précisé au chapitre II, on passerait rapidement notre temps à évaluer des individus représentant un maillage topologiquement non réalisable c'est-à-dire dont au moins un tétraèdre est retourné. On s'est donc tourné vers une approche **EP's (Evolution Program's,** cf. Chapitre II):

- les composantes sont exprimées dans leur codage naturel c'est-à-dire le codage réel.
- les opérateurs de croisement et surtout de mutation sont redéfinis.

Pour éviter les retournements de tétraèdres on utilisera deux artifices :

1/ On calculera initialement, pour chaque noeud, une **sphère maximale** de déplacement. Le rayon de la sphère est déterminé de façon heuristique en fonction de la distribution des longueurs des arêtes sortant du noeud (Fig. IV.28). Les opérateurs seront implantés pour respecter le mieux possible cette contrainte.
En pratique, pour construire les **sphères initiales**, on calcule, pour chaque noeud, la taille de la plus petite arête sortant de ce noeud puis, on la multiplie par un certain facteur $\alpha_{sph.}$ pour obtenir le rayon de la sphère. Ce facteur est aussi un paramètre de la méthode et il pourra, suivant la densité de mailles de la zone où se trouve le noeud considéré, être pris supérieur à 1, pour définir un rayon de sphère supérieur à l'arête minimum ou, au contraire, être pris inférieur à 1 pour obtenir un rayon initial inférieur à la taille de l'arête minimum. Si l'utilisateur choisit un rayon très supérieur à l'arête de plus petite taille il court le risque de générer un nombre important d'individus représentant des maillages non réalisables.

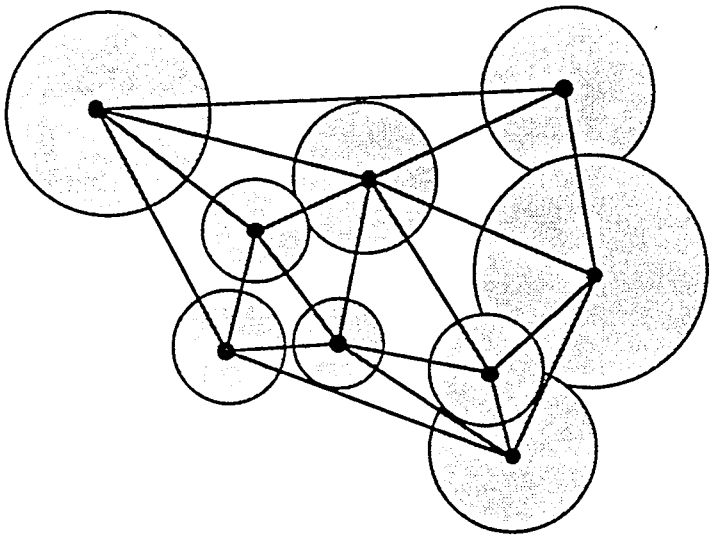


Figure IV.28: Sphères initiales (en 2 dimensions)

Cependant, compte tenu du déplacement dynamique de tous les noeuds, les opérateurs ne garantiront pas à 100% la réalisabilité du maillage issu d'un crossover ou d'une mutation. C'est pour cela qu'on utilise un second artifice à base de pénalités.

2/ On diminuera la qualité des éléments dont le volume est négatif. Ainsi, les individus présentant des tétraèdres de volume négatif seront pénalisés dans l'évaluation et auront moins de chance de survivre. La pénalité choisie possède les caractéristiques suivantes :

- Elle s'applique sur tous les éléments de volume négatif.
- Elle croît avec la valeur de l'itération courante.
- Elle croît avec le nombre d'éléments dans le maillage.
- Elle croît avec la valeur absolue du volume de la maille.

Dans notre étude nous avons utilisé le critère de [Talon89][FLUX3D] auquel nous avons associé une pénalité pour un tétraèdre de volume négatif de sorte que C devient:

$$C = 1. - \frac{1}{C^+} \frac{NBRMA}{\lambda} \frac{NGEN - 1}{NBGEN - 1}$$

où:

$$C^+ = \left| \frac{\text{Volume de la Sphère Inscrite}}{\text{Volume du Tétraèdre}} \right|$$

c'est-à-dire la valeur du critère si le tétraèdre n'était pas retourné.

NGEN: la génération courante.
NBGEN: le nombre maximum de générations.
NBRMA: le nombre de mailles dans le maillage.
λ: un coefficient >1 réglable suivant l'importance que l'utilisateur veut donner au terme *NBRMA*.

Ceci nous assure, qu'en fin d'exécution, le meilleur individu dans la population représentera un maillage topologiquement réalisable.
Il faut noter que certains individus posséderont des évaluations négatives mais ceci ne pose pas de problèmes puisque la sélection utilisée se fait par le rang (cf. Chapitre II).

Les opérateurs de croisement et de mutation sont au nombre de six. Les croisement permettent l'échange d'informations sur les déplacements des noeuds entre 2 individus (donc entre 2 maillages) et les mutations perturbent les vecteurs déplacements de certains noeuds d'un individu. Les opérateurs implantés sont décrits ci-dessous.

Croisement 1- point :

Contrairement aux croisements 1-point des chapitres précédents, le point de coupe est ici choisi aléatoirement entre 2 vecteurs de déplacements (Fig. IV.29). Il ne s'effectue en aucun cas entre les composantes d'un même noeud. Cela permet d'éviter des variations trop importantes d'une des composante d'un noeud qui pourrait déboucher sur un "individu non réalisable". L'intérêt de ce croisement est d'échanger de l'information sur le positionnement des noeuds entre différentes zones du maillage.

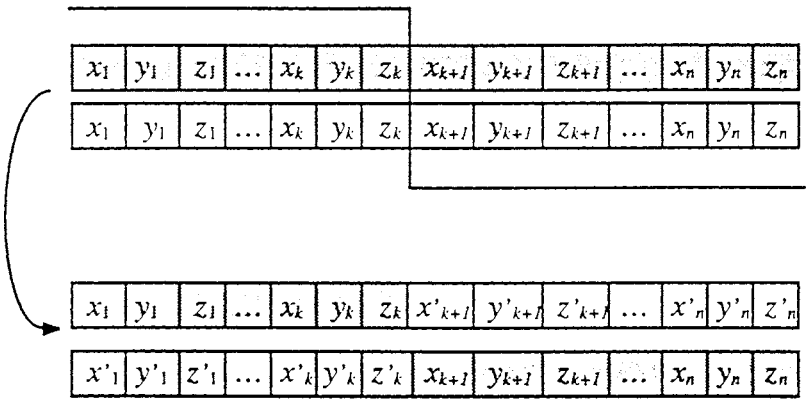


Figure IV.29: Croisement 1-point pour le maillage

Croisement 2-points :

Cet opérateur est identique au croisement 1-point mais 2 points de coupes sont sélectionnés le long de la chaîne

Croisement uniforme :

Cet opérateur échange des vecteurs déplacements de 2 individus avec une probabilité prise égale à 0,5.

Croisement arithmétique :

Ici, on génère aléatoirement un réel a entre 0 et 1 et on crée deux nouveaux individus s'_1 et s'_2 à partir de deux individus s_1 et s_2 de sorte que :

$$s'_1 = a s_1 + (1-a) s_2$$
$$s'_2 = (1-a) s_1 + a s_2$$

Cet opérateur calcule donc des moyennes arithmétiques aléatoires entre les vecteurs déplacements de chaque noeud des deux individus concernés.

Mutation 1:

Cette mutation agit sur un seul individu: elle perturbe aléatoirement un certain nombre de vecteurs de déplacements en leur affectant de nouvelles composantes. Elle garantit que les nouveaux vecteurs générés appartiennent bien à leur sphère de contraintes respectives. En fait, plusieurs mutations de ce type peuvent être implantées: elles dépendent de la méthode utilisée pour générer les nouveaux

vecteurs déplacements dans la sphère. Dans notre algorithme nous en avons implanté trois: deux d'entre elles agissent sur les trois composantes à la fois et la dernière modifie plus spécifiquement une des composantes du vecteur déplacement.
On peut remarquer que les opérateurs ci-dessus garantissent, la plupart du temps, que les vecteurs déplacements resteront dans leurs sphères de contraintes construites initialement. Cependant, des déplacements importants pour des noeuds voisins entraîneront à coup sûr des retournements de tétraèdres. C'est pour cette raison qu'un système de pénalités dans l'évaluation est utilisé.

Mutation 2 :

Il s'agit, pour l'individu sélectionné, de subir sur certains de ses noeuds, choisis aléatoirement, des déplacements de type bougé de points c'est-à-dire que le noeud est déplacé aléatoirement (et non par dichotomie) dans la direction d'un point supposé idéal ou en direction de l'isobarycentre de ses noeuds voisins (Fig. IV.30).

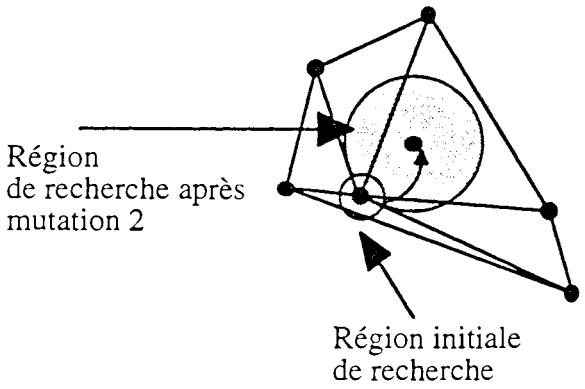


Figure IV.30: Effet d'une mutation 2 sur un des noeuds

La sphère de recherche de ce noeud est ensuite reconstruite pour permettre des explorations plus importantes.
Cet opérateur n'est en aucun cas déterministe puisque, d'une part les positions des noeuds voisins (pour l'individu concerné) ne sont pas figées et d'autre part, le nouveau vecteur déplacement est **aléatoirement** calculé dans la direction du point supposé idéal.

Les autres opérateurs de l'algorithme génétique ne sont pas spécifiques au maillage: nous avons employé une sélection linéaire par rang ainsi qu'une roue de Loterie pour choisir les opérateurs. Les poids des mutations et des croisements sont interpolés linéairement tout au long de l'exécution. On accordera toujours des poids plus importants:

- aux opérateurs de croisement en début d'exécution.
- aux opérateurs de mutations en fin d'exécution.

Ces poids sont des paramètres de l'algorithme.

3.4 Tests effectués

Dans les tests qui suivent nous avons utilisé le critère de [Talon89] implanté dans [FLUX3D] et une fonction objectif qui attribut la même importance aux différentes mailles.
Le premier maillage tétraédrique est celui de la machine synchrone à aimants (Fig. IV.10). Il est constitué de 8 volumes qui ont tous été maillés par le mailleur automatique de FLUX3D. La table IV.4 montre la répartition des éléments dans ces différents volumes et nous avons fait également figurer le nombre d'éléments possédant au moins un noeud non frontalier (les seuls susceptibles d'être modifiés par un algorithme de modification non topologique).

Volume	Nombre d'éléments	Nombre d'éléments ayant au moins un noeud à l'intérieur du volume	Nombre de noeuds régularisables
1	*3229*	*2117*	*172*
2	*457*	*360*	*25*
3	*267*	*213*	*11*
4	*473*	*391*	*28*
5	*788*	*155*	*6*
6	*249*	*118*	*6*
7	*333*	*199*	*11*
6	*245*	*182*	*9*
Total	*6041*	*3735*	*268*

Table IV.4: Répartition initiale des mailles dans les volumes pour le premier problème

Nous avons optimisé une première fois le maillage à l'aide d'un algorithme de modification topologique de type descendant en passant une à une les arêtes libres du maillage et en effectuant les opérations **3T<->2T**. La table IV.5 et la figure IV.31 montrent le nombre de mailles (**nm**) et le pourcentage (**%**) des mailles en fonction de la valeur du critère de qualité **Q** avant et après optimisation topologique. On constate que les moins bons éléments sont éliminés par cet algorithme. Comme nous nous intéressons aux algorithmes d'optimisation non topologique, nous avons fait figurer dans les table IV.6 et la figure IV.32 uniquement la répartition de la qualité pour les éléments possédant au moins un noeud non frontalier. Les conclusions restent identiques c'est-à-dire qu'après optimisation topologique les moins bons éléments disparaissent. On s'aperçoit aussi que la qualité moyenne du maillage s'améliore d'environ 7% (Table IV.7).

Qualité	%	nm	%	nm
0.9 < Q < 1.0	5.6%	340	6.5%	367
0.8 < Q < 0.9	17.6%	1061	20.7%	1161
0.7 < Q < 0.8	19.2%	1161	22.9%	1286
0.6 < Q < 0.7	16.4%	993	18.2%	1019
0.5 < Q < 0.6	12.2%	734	12.0%	675
0.4 < Q < 0.5	10.2%	615	9.4%	526
0.3 < Q < 0.4	7.2%	433	5.8%	328
0.2 < Q < 0.3	4.8%	292	3%	166
0.1 < Q < 0.2	3.4%	206	0.9%	52
0.0 < Q < 0.1	3,4%	206	0,5%	29
Totaux	100%	6041	100%	5609

Table IV.5: Histogramme du maillage avant et après optimisation topologique

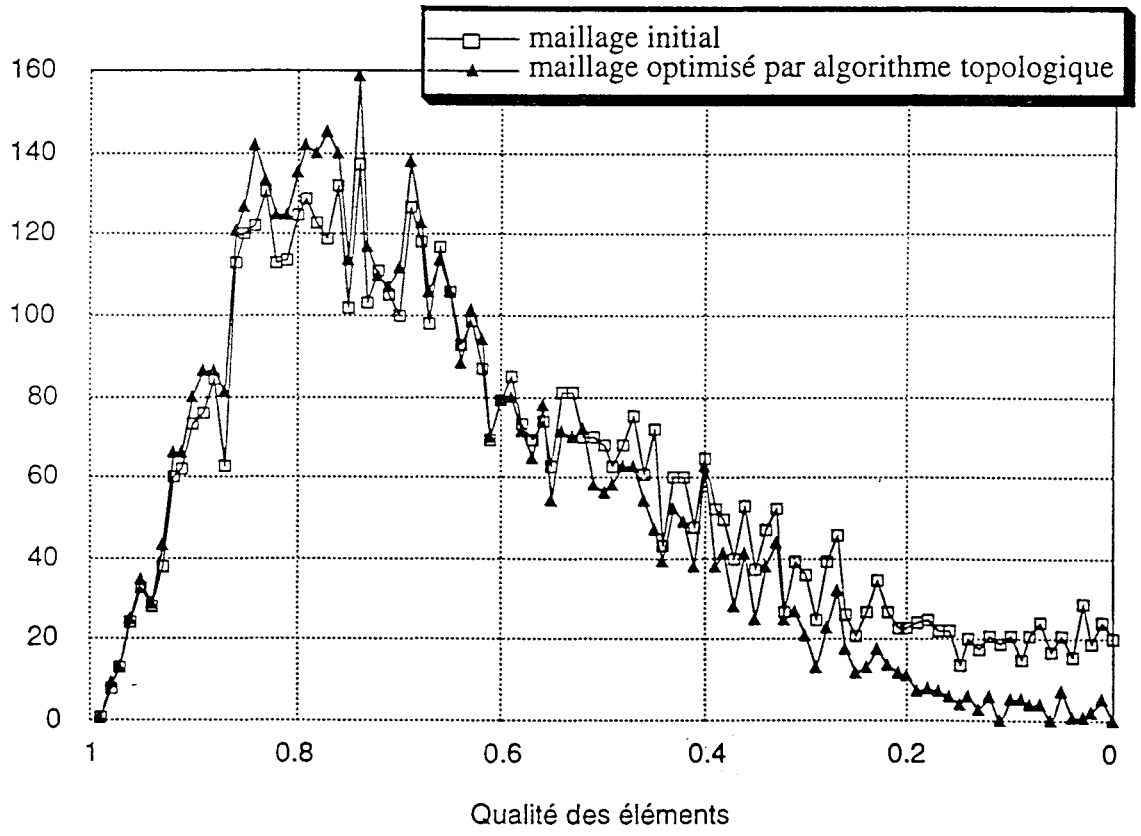


Figure IV.31: Répartition de la qualité de tous les éléments avant et après optimisation topologique

Qualité	%	n	%	n
0.9 < Q < 1.0	7.0%	264	8.5%	290
0.8 < Q < 0.9	19.8%	741	24.4%	831
0.7 < Q < 0.8	20.5%	765	24.8%	844
0.6 < Q < 0.7	16.3%	607	18.1%	617
0.5 < Q < 0.6	10.9%	408	10.8%	368
0.4 < Q < 0.5	8.1%	303	6.6%	226
0.3 < Q < 0.4	5.6%	211	3.5%	118
0.2 < Q < 0.3	4.5%	169	2%	69
0.1 < Q < 0.2	3.7%	138	1.1%	36
0.0 < Q < 0.1	3,5%	129	0,3%	9
Totaux	100%	3735	100%	3408

Table IV.6: Histogramme du maillage avant et après optimisation topologique pour les éléments ayant au moins un noeud non frontalier

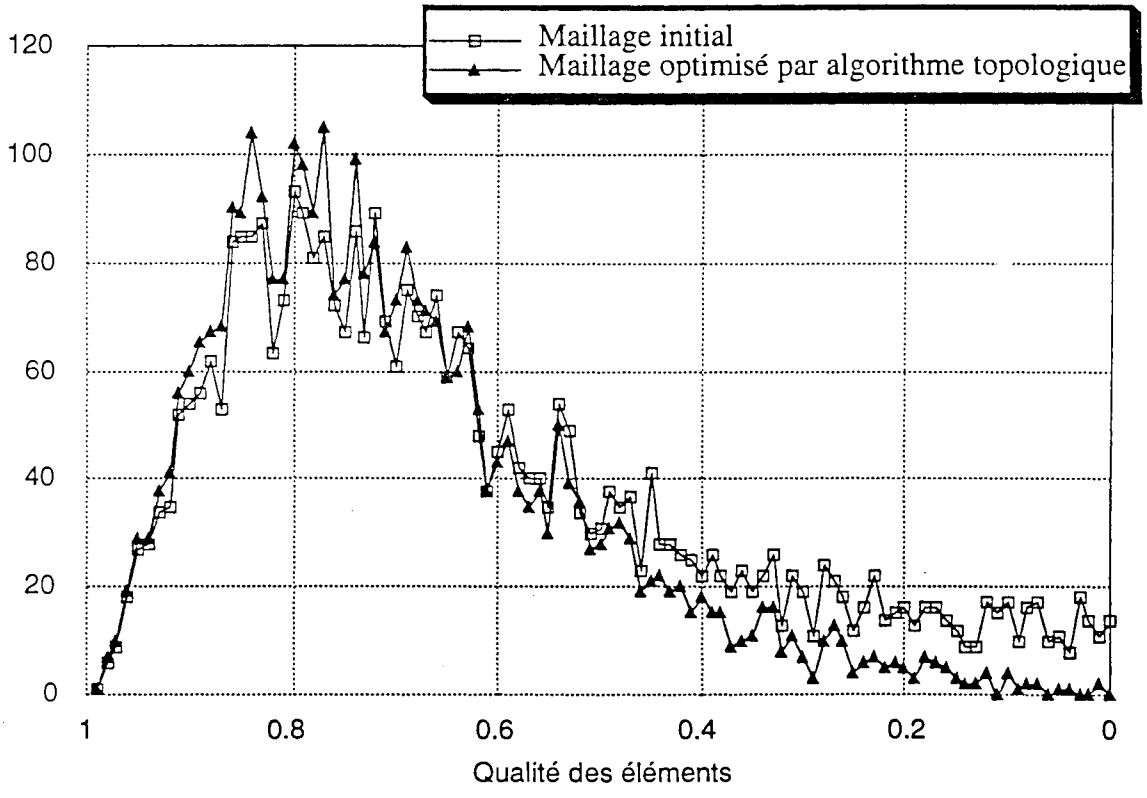


Figure IV.32: Répartition de la qualité des éléments possédant au moins un noeud non frontalier avant et après optimisation topologique

	Initiale	*Topologique*
Tous les éléments	*0.61034*	*0.6663*
Eléments avec un noeud non frontalier	*0.6301*	*0.6999*

Table IV.7: Qualité moyenne du maillage avant et après optimisation topologique

Après cette optimisation topologique, nous avons effectué une optimisation non topologique à l'aide de l'algorithme génétique. Les résultats seront montrés pour les éléments possédant au moins un noeud non frontalier. Le paramètre multiplicatif $\alpha_{sph.}$ est pris constant quel que soit le noeud. Ci dessous les divers paramètres de l'algorithme génétique:

- NBPOP: Taille de la population.
- tsel: Valeur du paramètre de sélection par rang (cf. Chapitres I et II).
- $\alpha_{sph.}$: Facteur multiplicatif pour déterminer le rayon de la sphère initiale.
- λ : Paramètre entrant en compte dans la pénalité.
- P^i: Probabilités initiales des opérateurs de croisement et de mutation.
- P^f: Probabilités finales des opérateurs de croisement et de mutation.
- $\%N_{mutés}$: Pourcentage maximum de noeuds mutés pour une chaîne subissant un opérateur de mutation: la mutation perturbe un pourcentage aléatoire $\%N < \%N_{mutés}$ de noeuds dans la chaîne.

Dans la **première expérience** nous n'avons pas utilisé de mutation de type 2 et nous avons lancé l'algorithme génétique 2 fois consécutivement. Ainsi on espère que les positions initiales des noeuds pour la deuxième exécution permettront la construction de sphères de rayon plus important et donc une recherche plus importante sans augmentation du nombre d'individus non réalisables. Nous avons pris un facteur $\alpha_{sph.}$=1. pour la première exécution et $\alpha_{sph.}$=1.5 pour la seconde. Les autres paramètres sont montrés dans les tables IV.8, IV.9, IV.10.

NBPOP	tsel	λ	%Nmutés
50	1.7	20	10 %

Table IV.8: Paramètres de l'algorithme génétique de maillage

P^i	*0.1*	*0.1*	*0.1*	*0.25*
P^f	*0.05*	*0.05*	*0.1*	*0.15*

Table IV.9: Probabilités initiales et finales des opérateurs de croisement

P^i	*0.08*	*0.02*	*0.1*
P^f	*0.25*	*0.05*	*0.2*

Table IV.10: Probabilités initiales et finales des opérateurs de mutation (type 1 uniquement)

Les résultats obtenus sont présentés dans la table IV.11 où nous avons aussi fait figurer les résultats de l'optimisation topologique.

Qualité	%	n	%	n	%	n
0.9 < Q < 1.0	7.0%	264	8.5%	290	11.8%	402
0.8 < Q < 0.9	19.8%	741	24.4%	831	28.1%	958
0.7 < Q < 0.8	20.5%	765	24.8%	844	23.4%	799
0.6 < Q < 0.7	16.3%	607	18.1%	617	17%	581
0.5 < Q < 0.6	10.9%	408	10.8%	368	9.5%	325
0.4 < Q < 0.5	8.1%	303	6.6%	226	4.5%	153
0.3 < Q < 0.4	5.6%	211	3.5%	118	3%	101
0.2 < Q < 0.3	4.5%	169	2%	69	1.6%	56
0.1 < Q < 0.2	3.7%	138	1.1%	36	0.6%	20
0.0 < Q < 0.1	3,5%	129	0,3%	9	0,4%	13
Totaux	100%	3735	100%	3408	100%	3408

Table IV.11: Histogramme du maillage avant et après les optimisations topologique et topologique+AG

On constate que le nombre d'éléments de très bonne qualité augmente de façon sensible par rapport à une optimisation topologique seule et finalement la qualité moyenne du maillage s'améliore encore de 3% (Table IV.12).

Initiale	*Topologique*	*Topologique+AG*
0.6301	*0.6999*	*0.7278*

Table IV.12: Qualité moyenne du maillage avant et après les optimisations topologique et topologique+AG

Dans une **seconde expérience** nous avons ajouté une mutation de type 2 et nous n'avons effectué qu'une seule exécution de l'algorithme génétique. En effet, cet opérateur possède la capacité de réaliser des déplacements beaucoup plus important sur les noeuds sans pour autant augmenter le nombre d'éléments "retournés" (puisqu'il déplace a priori le noeud vers l'intérieur de l'enveloppe formé par ses voisins). Les résultats obtenus sont montrés dans les tables IV.13 et IV.14.

Qualité	%	n	%	n	%	n
0.9 < Q < 1.0	7.0%	264	8.5%	290	13.1%	445
0.8 < Q < 0.9	19.8%	741	24.4%	831	29.5%	1004
0.7 < Q < 0.8	20.5%	765	24.8%	844	22.4%	764
0.6 < Q < 0.7	16.3%	607	18.1%	617	16.8%	571
0.5 < Q < 0.6	10.9%	408	10.8%	368	8.8%	301
0.4 < Q < 0.5	8.1%	303	6.6%	226	4.6%	156
0.3 < Q < 0.4	5.6%	211	3.5%	118	2.7%	92
0.2 < Q < 0.3	4.5%	169	2%	69	1.5%	50
0.1 < Q < 0.2	3.7%	138	1.1%	36	0.5%	16
0.0 < Q < 0.1	3,5%	129	0,3%	9	0,3%	9
Totaux	100%	3735	100%	3408	100%	3408

Table IV.13: Histogramme du maillage lorsque l'AG possède une mutation de type 2

Initiale	*Topologique*	*Topologique+AG*
0.6301	*0.6999*	*0.7370*

Table IV.14: Qualité moyenne du maillage lorsque l'AG possède une mutation de type 2

Les résultats, quoique sensiblement équivalents à ceux de la première expérience, sont meilleurs avec cet opérateur de mutation. En effet, la qualité moyenne est supérieure de 1% dans cette expérience et le nombre d'éléments de grandes qualité augmente encore.
Au vus de ces expériences, **l'algorithme topologique+l'algorithme non topologique génétique** ont permis d'**améliorer la qualité** moyenne du maillage de près de **10%** et surtout d'**éliminer les éléments de très mauvaise qualité.**

En ce qui concerne les temps de calcul, l'optimum (global a priori) a été localisé en 12 000 évaluations de l'objectif, soit près de 36 000 000 d'évaluations de la qualité d'un élément (C), et a nécessité près de 8 h sur un HP700/80.

Nous avons voulu comparer les résultats obtenus par l'algorithme génétique avec l'algorithme de bougé de points de [Henot93] qui peut être vu comme un algorithme déterministe puisque les noeuds sont bougés 1 à 1 et non de manière globale comme pour l'AG. Nous avons lancé deux fois consécutivement l'algorithme de bougé de points sur le même problème. Les résultats sont présentés dans les tables IV.15 et IV.16. On constate que la distribution des éléments et la valeur moyenne sont sensiblement identiques à celles obtenues par l'algorithme génétique. Ceci nous amène à nous

demander:

- 1 Si la fonction objectif, qui dépend de la répartition des noeuds dans le maillage, est une fonction multi-optima mais avec des valeurs sensiblement identiques.

- 2 Si cette même fonction possède un seul optimum qui serait alors atteint à la fois par une méthode globale (l'AG) et par une méthode déterministe (en l'occurrence le bougé de points).

Pour répondre à ces interrogations nous avons calculé l'écart relatif entre l'optimum atteint par l'algorithme génétique dans les 2 expériences précédentes et l'optimum atteint par le bougé de points (Fig. IV.33 et IV.34). Ces courbes ont été obtenues sur les 172 noeuds non frontaliers du premier volume (l'air entourant le dispositif). On constate que les résultats sont très proches: 6% de différence au maximum sur l'un des noeuds dans la première expérience et 2,5% sur l'un des noeuds dans la seconde. Il semble donc bien que le critère utilisé soit une fonction possédant un seul optimum et que la répartition optimale des noeuds soit trouvée à la fois par un algorithme de bougé de points et par un algorithme génétique. Cependant, comme nous l'avions vu dans les deux premiers chapitres, le nombre d'évaluations nécessaire pour une méthode génétique est bien plus important que pour une méthode déterministe (270 000 évaluations de C soit 3mn sur HP700/80). Ces expériences suggèrent aussi l'emploi de cet algorithme de Bougé de Points comme algorithme non topologique optimum ce qui n'est pas le cas de l'algorithme d'isobarycentrage (Fig. IV.35) qui n'a pas pu localiser l'optimum.

Qualité	%	n	%	n	%	n
0.9 < Q < 1.0	7.0%	264	8.5%	290	14.1%	480
0.8 < Q < 0.9	19.8%	741	24.4%	831	28.9%	986
0.7 < Q < 0.8	20.5%	765	24.8%	844	22.9%	782
0.6 < Q < 0.7	16.3%	607	18.1%	617	16.3%	556
0.5 < Q < 0.6	10.9%	408	10.8%	368	8.5%	289
0.4 < Q < 0.5	8.1%	303	6.6%	226	4.6%	157
0.3 < Q < 0.4	5.6%	211	3.5%	118	2.4%	83
0.2 < Q < 0.3	4.5%	169	2%	69	1.4%	47
0.1 < Q < 0.2	3.7%	138	1.1%	36	0.6%	19
0.0 < Q < 0.1	3,5%	129	0,3%	9	0,3%	9
Totaux	100%	3735	100%	3408	100%	3408

Table IV.15: Histogramme du maillage avant et après les optimisations topologique et topologique+Bougé de Points

Initial	Topologique	Topologique + Bougé de points
0.6301	0.6999	0.7381

Table IV.16: Qualité moyenne du maillage avant et après les optimisations topologique et topologique +Bougé de points

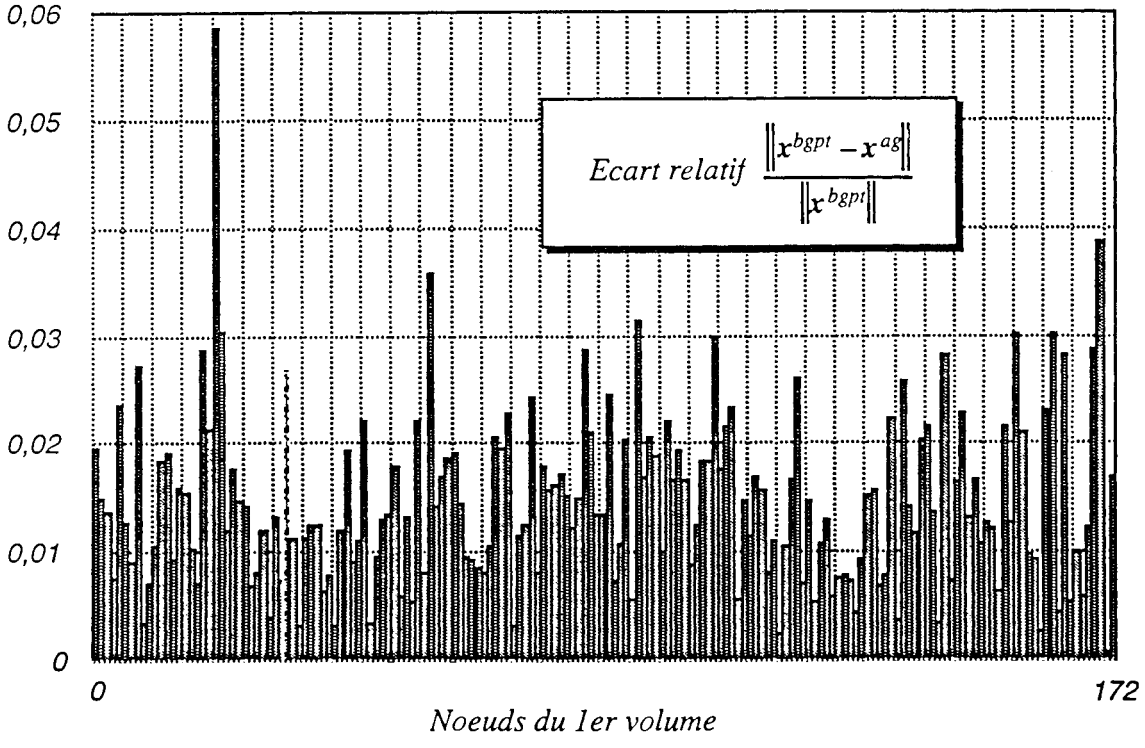


Figure IV.33: Ecart relatif entre l'optimum atteint par l'AG dans la 1ere expérience et l'optimum atteint par le Bougé de points

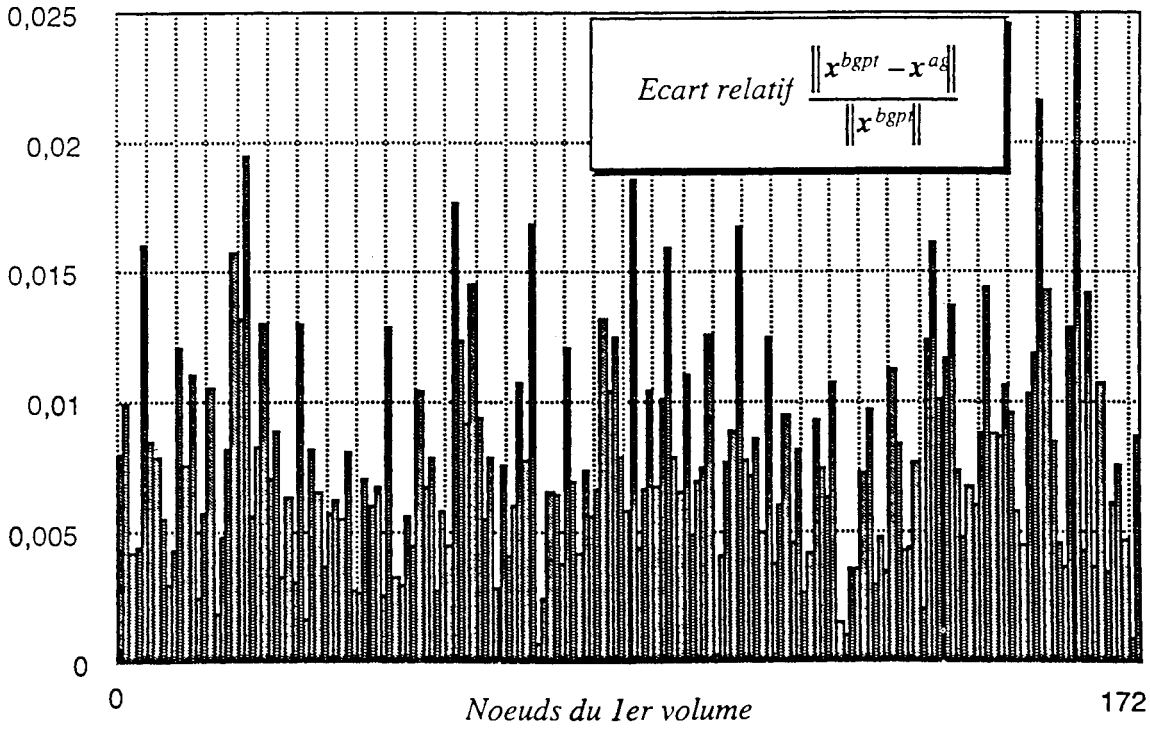


Figure IV.34: Ecart relatif entre l'optimum atteint par l'AG dans la 2eme expérience et l'optimum atteint par le Bougé de points

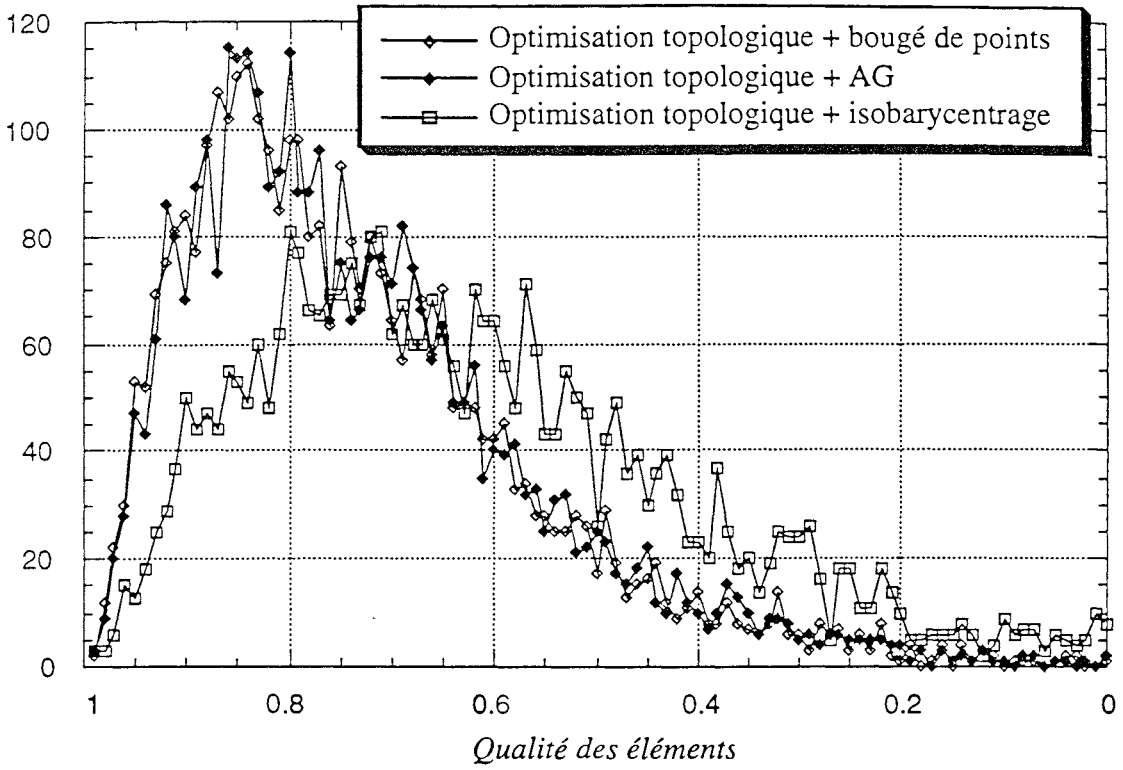


Figure IV.35: Comparaison des optimisations non topologiques

Nous avons réalisé la même confrontation sur un deuxième problème (Fig. IV.36). Les valeurs moyennes sont présentées dans la table IV.17 et l'écart relatif pour les 230 noeuds du volume 3 (l'air entourant le dispositif) entre les 2 méthodes est montré dans la figure IV.37. On constate encore une fois que l'optimum atteint par les deux méthodes non topologique est identique.

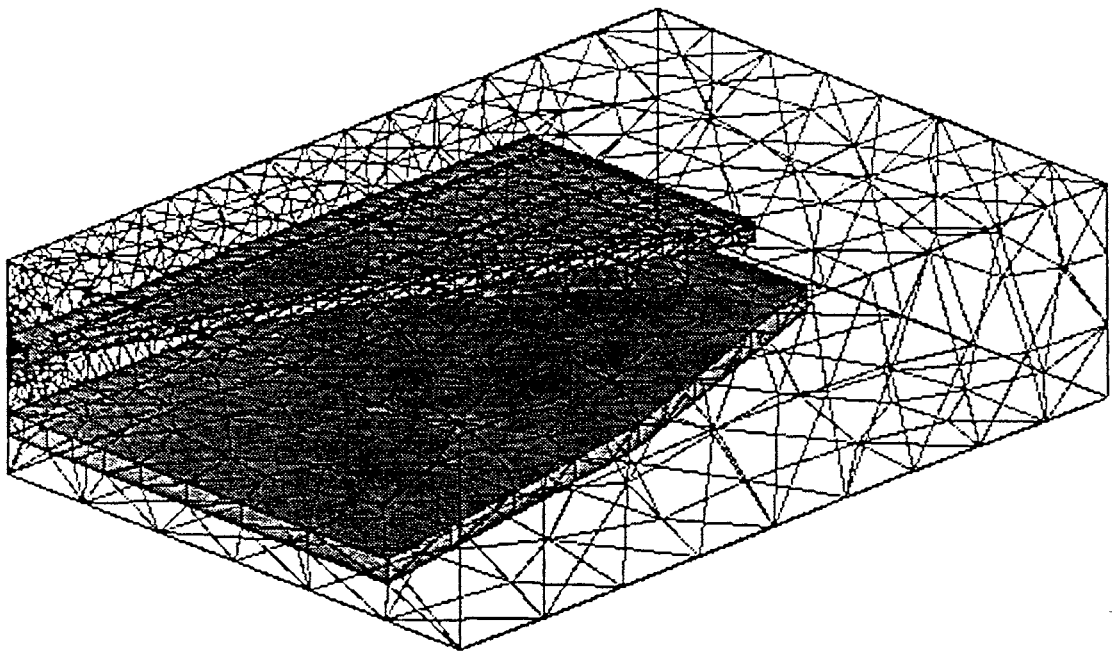


Figure IV.36: Maillage initial du problème numéro 2

Initial	*Topologique*	*Topologique + AG*	*Topologique + Bougé de points*
0.630	*0.690*	*0.720*	*0.720*

Table IV.17: Qualités moyennes pour les éléments de l'air entourant le dispositif (3742 initialement, 3464 après optimisation topologique)

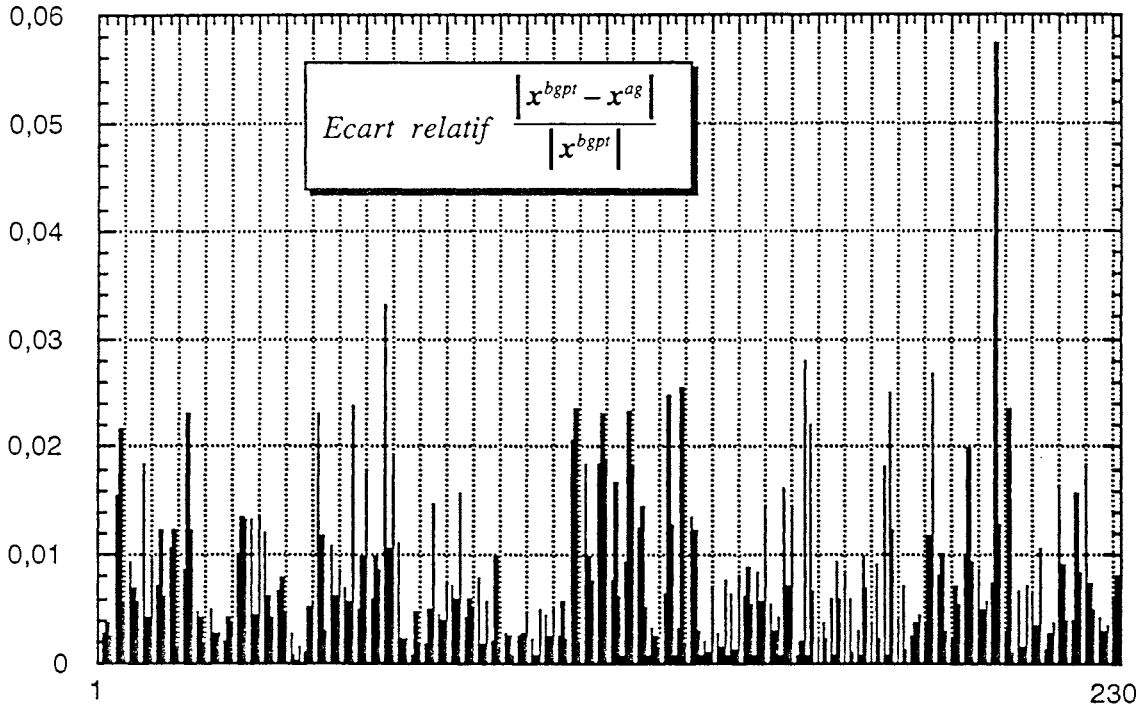


Figure IV.37: Ecart relatif entre l'optimum atteint par l'AG et l'optimum atteint par le Bougé de points pour le deuxième problème

Nous avons donc montré qu'un algorithme globalement convergent (l'AG) a atteint le même optimum que l'algorithme de Bougé de points proposé par [Henot93]. Ceci semble indiquer que le Bougé de points est un algorithme performant et que la fonction objectif ne possède qu'un optimum (donc global !) ce qui n'était pas assuré a priori. D'ailleurs pour confirmer ceci, nous avons réalisé un bougé de points en ne prenant plus les noeuds 1 à 1 dans u*n ordre bien déterminé* mais en choisissant à chaque fois un noeud tiré *aléatoirement.* L'algorithme de bougé de points a été relancé 2 fois consécutivement pour chaque essai. On s'aperçoit (Fig. IV.38) que les configurations de noeuds atteintes sont identiques pour chacun des essais ce qui semble indiquer que l'algorithme de Bougé de points donne la même configuration des noeuds quel que soit l'ordre de parcours de ces noeuds.

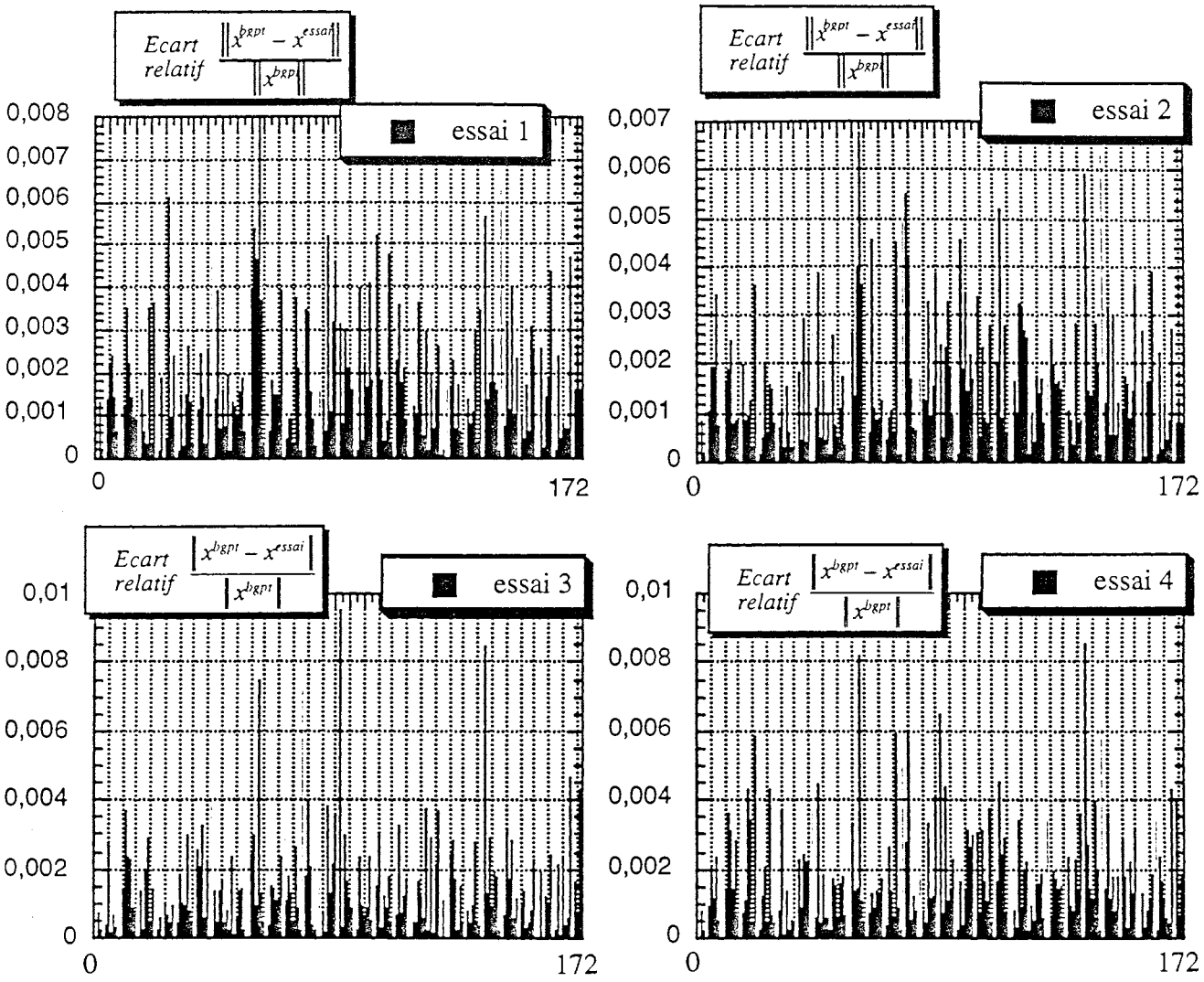


Figure IV.38: Ecart relatif entre différents bougés de points aléatoires

4 CONCLUSION

Dans ce chapitre, nous avons travaillé sur l'optimisation *a priori* des éléments tétraédriques. Cette optimisation est nécessaire car la qualité géométrique des éléments générés par un mailleur automatique de type Delaunay n'est pas contrôlable par l'utilisateur. Nous avons d'abord montré que le critère de qualité utilisé par le logiciel FLUX3D rendait effectivement bien compte de la forme géométrique de l'élément. Nous avons ensuite brièvement décrit les différentes techniques d'optimisation de maillage qui existent à l'heure actuelle. Nous avons séparé celles ci en deux familles: les algorithmes d'amélioration topologique et les algorithmes d'amélioration non topologique. Les premiers jouent sur les arêtes et les faces du maillage alors que les seconds s'occupent uniquement de modifier les coordonnées des noeuds du maillage et il apparaît indispensable, comme l'avaient souligné certains auteurs, de procéder à une optimisation topologique avant de lancer un algorithme de modifications non topologique.

Finalement, nous nous sommes penchés sur un algorithme non topologique génétique. En effet, il semble nécessaire d'utiliser un algorithme de recherche globale car rien ne nous indique a priori que la disposition des noeuds dans le maillage obtenue par les algorithmes non topologiques de type isobarycentrage ou Bougé de points soit la meilleure possible. L'algorithme génétique présenté ici s'éloigne de l'algorithme de Holland puisqu'il n'utilise ni codage binaire ni opérateurs classiques de croisement et de mutation. Nous nous sommes écartés de l'algorithme génétique classique en raison du nombre très important de discontinuités (retournements de tétraèdres) qui seraient apparues si le codage binaire avait été employé. L'approche génétique faite ici est donc de type EP's (cf. Chapitre II): les opérateurs ont été redéfinis spécialement pour le maillage et en vue d'une optimisation non topologique.

Enfin, nous avons lancé l'algorithme génétique sur certains problèmes tests. L'algorithme a effectivement localisé un optimum qui a permis d'obtenir un gain de 10% sur la qualité moyenne du maillage ainsi que l'élimination des éléments de très mauvaises qualité.

Nous avons ensuite comparé cet optimum à celui obtenu par l'algorithme de Bougé de points de [Henot93]. Il est apparu que ces optima étaient très proches l'un de l'autre ce qui signifie que le critère de qualité du maillage est une fonction unimodale et que l'algorithme de Bougé de points est un algorithme déterministe (on arrive toujours à la même configuration finale) efficace.

Chapitre V

Optimisation de forme de structures électromagnétiques

Chapitre V

Optimisation de forme de structures électromagnétiques

1 INTRODUCTION ET PRESENTATION DU PROBLEME

Dans cette partie, nous nous proposons de traiter l'optimisation de forme de structures électromagnétiques. Il s'agit de trouver les "bonnes" valeurs des paramètres de conception d'un dispositif (géométriques et/ou physiques) afin de répondre à un certain nombre d'objectifs tout en satisfaisant des contraintes mécaniques ou physiques sur ces paramètres. Cette amélioration peut être réalisée manuellement par l'ingénieur mais ceci implique une bonne connaissance de la sensibilité des paramètres par rapport au problème traité. Par ailleurs, ce type de démarche est difficilement généralisable et demande un investissement important en temps de calcul sans aucune garantie que la solution obtenue soit effectivement la meilleure possible.

Le but ici est au contraire d'utiliser conjointement une méthode d'optimisation automatique et efficace avec une méthode de calcul de champs précise.

En électromagnétisme, la fonction objectif que l'on cherche à optimiser dépend en général de grandeurs qui ne peuvent pas être obtenues analytiquement et on a alors recourt à une méthode numérique pour les calculer. Au Laboratoire, nous disposons des logiciel Flux2d (en 2 dimensions) et Flux3d (en 3 dimensions) qui s'appuient sur la méthode des éléments finis (que nous avons brièvement évoquée au chapitre IV) pour calculer les grandeurs électromagnétiques. On peut donc réaliser le couplage entre un de ces logiciels et un des algorithmes d'optimisation vus aux chapitres I et II. A cette fin, il est nécessaire de mettre en œuvre des outils pour le calcul des dérivées lorsqu'un algorithme de type gradient est utilisé.

Cependant, le calcul des grandeurs électromagnétiques par un logiciel éléments finis génère des temps très importants. En effet, chaque jeu de nouvelles valeurs des paramètres nécessite une analyse éléments finis complète c'est-à-dire un maillage, un assemblage et la résolution d'un système linéaire pour calculer la valeur de la variable d'état (qui représente le phénomène étudié) sur tout le domaine. L'utilisation d'un algorithme génétique avec cette méthode risque donc de perdre de son intérêt du fait du grand nombre d'évaluations de l'objectif nécessaire pour atteindre l'optimum. Dans le cadre de problèmes linéaires et bidimensionnels, pour diminuer de façon sensible les coûts des évaluations, nous avons utilisé une approximation de la variable d'état qui se base sur un développement de Taylor de la solution éléments finis obtenue pour des valeurs initiales des paramètres.

Dans ce chapitre, nous allons voir comment réaliser ce couplage original et nous validerons cette démarche sur un problème test de bobines supraconductrices. Nous commencerons cependant par une

description des calculs de champs dans un logiciel éléments finis que nous présenterons dans le cadre d'un problème magnétodynamique bidimensionnel.

2 PRINCIPE DE LA METHODE DES ELEMENTS FINIS

2.1 Du phénomène physique étudié au modèle

Nous avons déjà décrit, de manière succincte, la méthode des éléments finis dans le chapitre IV. Il s'agit ici de préciser quelques points concernant le calcul des grandeurs électromagnétiques. Le lecteur intéressé par une présentation plus complète pourra se reporter aux ouvrages de [Sabonnadière86][Touzot84].
Nous nous plaçons dans le domaine de l'électromagnétisme basse fréquence, ce qui permet de négliger les courants de déplacements. Dans ce cadre, les équations de Maxwell se résument à :

$$rot\, H = J \tag{V.1}$$

$$div\, B = 0 \tag{V.2}$$

$$rot\, E = -\frac{\partial B}{\partial t} \tag{V.3}$$

E: champ électrique (V/m)
H: champ magnétique (A/m)
B: induction magnétique (T)
J: densité de courant (A/m²)

A cet ensemble d'équations il faut ajouter les équations constitutives qui lient ces différentes grandeurs aux caractéristiques physiques des matériaux :

$$B = \mu H + B_r \tag{V.4}$$

$$J = J_S + \sigma E \tag{V.5}$$

J_S: densité des courants d'excitation qui alimentent le dispositif (A/m²)
μ : perméabilité magnétique (H/m)
σ : conductivité électrique (S/m)
B_r: induction rémanente (T)

Suivant les matériaux utilisés, les grandeurs μ et σ sont soit des scalaires (matériaux isotropes), soit des tenseurs modélisant alors le comportement anisotrope des matériaux employés. Le champ rémanent B_r est nul sauf en présence d'aimants.
Les deux grandeurs physiques inconnues sont le champ magnétique H et l'induction magnétique B qui peuvent être calculés en introduisant une variable d'état A, appelée potentiel vecteur magnétique et tel que :

$$B = rot\, A \tag{V.6}$$

De plus, pour définir de manière unique A, il faut également fixer la valeur de sa divergence. On ajoute alors la condition (appelée *jauge de Coulomb*):

$$div\,A = 0 \tag{V.7}$$

En utilisant cette grandeur dans l'équation (V.1) et en supposant B_r nul et les matériaux isotropes les équations de Maxwell se réduisent à :

$$rot(\nu\, rot\, A) = J \tag{V.8}$$

où ν désigne la réluctivité magnétique ($\nu = 1/\mu$)
Finalement grâce aux équations (V.4) et (V.5) on obtient :

$$rot(\nu\, rot\, A) + \sigma\frac{\partial A}{\partial t} = J_S \tag{V.9}$$

Les équations (V.9) et (V.7) permettent de définir complètement A. De plus, pour des objets suffisamment longs pour négliger les effets d'extrémité ou possédant une symétrie de révolution, les courants sont généralement perpendiculaires au plan d'étude. Dans ce cas, le potentiel vecteur A et le courant d'excitation J_S n'ont plus qu'une seule composante utile et la *jauge de Coulomb* est implicitement vérifiée. Pour des problèmes linéaires et un courant d'excitation de pulsation ω , on peut prendre une notation complexe pour A et J_S :

$$A = \begin{bmatrix} 0 \\ 0 \\ Re\left(A(x,y)e^{j\omega t}\right) \end{bmatrix} \text{ et } J_S = \begin{bmatrix} 0 \\ 0 \\ Re\left(J_S(x,y)e^{j\omega t}\right) \end{bmatrix} \tag{V.10}$$

où $Re()$ désigne la partie réelle et j^2=-1.

L'équation (V.9) devient alors:

$$-\nabla(\nu\nabla A) + j\omega\sigma A = J_S \tag{V.11}$$

Nous sommes donc amenés à résoudre une équation aux dérivées partielles (EDP) pour déterminer le potentiel vecteur A à partir duquel on pourra ensuite calculer l'induction B et le champ H. A l'équation (V.11) on ajoute des conditions aux limites sur les frontières du domaine du dispositif étudié. Les conditions aux limites les plus courantes sont la condition de Dirichlet ($A=A_0$) sur une partie Γ_1 de la frontière et la condition de Neumann homogène ($\frac{\partial A}{\partial n} = 0$) donnée sur le reste de la frontière Γ_2. Ces conditions permettent de définir les symétries du dispositif.

2.2 De l'EDP au système matriciel- Méthode de Galerkin

On ne sait pas actuellement trouver une solution analytique aux équations aux dérivées partielles du type (V.11) pour des problèmes réels. Pour cette raison, on utilise des méthodes numériques qui ramènent le problème continu à un problème discret. C'est en particulier le cas de la méthode des éléments finis que nous allons décrire plus en détail qu'au chapitre IV où seul l'aspect du maillage avait été abordé.

La méthode des éléments finis consiste à découper le domaine d'étude Ω en sous-domaines de formes géométriques simples puis, à l'aide d'une formulation intégrale de (V.11), elle transforme le système d'EDP en un système d'équations linéaires plus simples à traiter. Les formulations intégrales les plus couramment employées pour le calcul des champs électromagnétiques sont la méthode de Galerkin et la formulation variationnelle. Nous présenterons uniquement la méthode de Galerkin appelée aussi méthode des résidus pondérés.

On cherche $A(x,y)$ une fonction continue, dérivable par morceaux sur Ω et vérifiant l'équation (V.11) ainsi que les conditions aux limites sur la frontière Γ du domaine.
En multipliant l'équation (V.11) par une fonction W_i continue, dérivable par morceaux et nulle sur Γ_1, on obtient après intégration sur Ω :

$$\iint_{\Omega}\left[-\nabla(\nu\nabla A)+j\omega\sigma A\right]W_i\,d\Omega=\iint_{\Omega}J_S W_i\,d\Omega \qquad \text{(V.12)}$$

soit en développant l'opérateur *nabla* :

$$-\iint_{\Omega}\left[\frac{\partial}{\partial x}\left(\nu\frac{\partial A}{\partial x}\right)+\frac{\partial}{\partial y}\left(\nu\frac{\partial A}{\partial y}\right)\right]W_i\,d\Omega+\iint_{\Omega}j\omega\sigma A W_i\,d\Omega=\iint_{\Omega}J_S W_i\,d\Omega \qquad \text{(V.13)}$$

L'opération qui vient d'être faite peut être vue comme un produit scalaire dans l'espace fonctionnel des fonctions continues et dérivables par morceaux. Ainsi, l'équation (V.13) traduit le fait que si deux fonctions sont égales alors leurs produits scalaires par W_i le sont aussi.
La méthode de Galerkin consiste à prendre n *fonctions tests* W_i pour lesquelles on écrit l'équation (V.13). En utilisant la *formule de Green*, on obtient :

$$\iint_{\Omega}\nu\left(\frac{\partial A}{\partial x}\frac{\partial W_i}{\partial x}+\frac{\partial A}{\partial y}\frac{\partial W_i}{\partial y}\right)d\Omega-\int_{\Gamma}\nu\frac{\partial A}{\partial n}W_i\,d\Gamma+\iint_{\Omega}j\omega\sigma A W_i\,d\Omega=\iint_{\Omega}J_S W_i\,d\Omega \qquad \text{(V.14)}$$

où n désigne la normale extérieure à Γ et compte tenu de W_i=0 sur Γ_1 et $\frac{\partial A}{\partial n}=0$ sur Γ_2, (V.14) se

réduit à :

$$\iint_\Omega \nu\left(\frac{\partial A}{\partial x}\frac{\partial W_i}{\partial x}+\frac{\partial A}{\partial y}\frac{\partial W_i}{\partial y}\right)d\Omega+\iint_\Omega j\omega\sigma A W_i\, d\Omega=\iint_\Omega J_S W_i\, d\Omega \qquad (V.15)$$

Si le domaine est discrétisé en N_e éléments finis et possède N_n noeuds de discrétisation, alors la fonction $A(x,y)$ peut être approximée sur tout le domaine à l'aide d'une combinaison linéaire du type :

$$A(x,y)=\sum_{j=1}^{N_n} A_j\varphi_j(x,y) \qquad (V.16)$$

où A_j désigne la valeur de A (inconnue) au noeud j de la discrétisation et $\varphi_j(x,y)$ est une fonction continue et dérivable par morceaux qui assure la liaison entre le problème discret (les A_j) et le problème continu dont on cherche une solution. En introduisant (V.16) dans l'équation (V.15) on a alors :

$$\iint_\Omega \nu\sum_{j=1}^{N_n} A_j\left(\frac{\partial\varphi_j}{\partial x}\frac{\partial W_i}{\partial x}+\frac{\partial\varphi_j}{\partial y}\frac{\partial W_i}{\partial y}\right)d\Omega+\iint_\Omega j\omega\sigma\sum_{j=1}^{N_n} A_j\varphi_j\, W_i\, d\Omega=\iint_\Omega J_S W_i\, d\Omega \qquad (V.17)$$

soit encore :

$$\forall i\ \sum_{j=1}^{N_n} A_j\left[\iint_\Omega \nu\left(\frac{\partial\varphi_j}{\partial x}\frac{\partial W_i}{\partial x}+\frac{\partial\varphi_j}{\partial y}\frac{\partial W_i}{\partial y}\right)d\Omega+\iint_\Omega j\omega\sigma\varphi_j\, W_i\, d\Omega\right]=\iint_\Omega J_S W_i\, d\Omega \qquad (V.18)$$

En prenant le nombre de fonctions tests égal au nombre de noeuds de la discrétisation, on obtient un système linéaire de N_n équations à N_n inconnues de la forme:

$$\sum_{j=1}^{N_n} M_{ij}A_j=S_i \qquad (V.19)$$

avec :

$$M_{ij}=\left[\iint_\Omega \nu\left(\frac{\partial\varphi_j}{\partial x}\frac{\partial W_i}{\partial x}+\frac{\partial\varphi_j}{\partial y}\frac{\partial W_i}{\partial y}\right)d\Omega+\iint_\Omega j\omega\sigma\varphi_j\, W_i\, d\Omega\right] \qquad (V.20)$$

$$S_i=\iint_\Omega J_S W_i\, d\Omega \qquad (V.21)$$

Cette méthode basée sur une formulation intégrale reste très générale. En fait, dans la méthode des éléments finis, on choisit les fonctions d'approximation $\varphi_j(x,y)$ égales aux fonctions tests $W_i(x,y)$ et possédant la propriété d'être polynomiale par morceaux. Il reste donc à calculer ces $\varphi_j(x,y)$ pour pouvoir déterminer complètement le système linéaire (V.19).

Sur chaque élément fini, la variable d'état peut être approximée à l'aide d'une combinaison linéaire du type:

$$A^{(e)}(x,y)=\sum_{i=1}^{N_{ne}} A_i \alpha_i^{(e)}(x,y) \qquad (V.22)$$

où N_{ne} est le nombre de noeuds de l'élément (e) et $\alpha_i^{(e)}(x,y)$ une fonction bilinéaire en x et y, appelée *fonction de forme* de l'élément (e), et qui vérifie:

$$\alpha_i^{(e)}(x_j,y_j)=\delta_{i,j}$$

c'est-à-dire que la fonction de forme numéro i est nulle pour les noeuds j de l'élément (e) qui ne sont pas égaux à i et prend la valeur 1 pour $j=i$.

Les fonctions d'approximation φ_j sont construites à partir de ces fonctions de forme sur chaque élément. φ_j est nulle si le sommet de numéro j n'appartient pas à l'élément (e) donné et est égale à la fonction $\alpha_j^{(e)}$ dans le cas contraire. Par construction, les fonctions d'approximation sont donc continues et dérivables sur chaque élément fini. Dans l'équation (V.20), M_{ij} représente une intégrale sur tout le domaine, mais avec le choix fait pour les fonctions φ_j, les éléments auxquels n'appartiennent pas les noeuds i et j ont une contribution nulle. Par conséquent M_{ij} peut s'écrire comme une somme de termes élémentaires :

$$M_{ij}=\sum_{(e)} M_{ij}^{(e)} \qquad (V.23)$$

avec :

$$M_{ij}^{(e)}=\left[\iint_{\Omega_e} \nu\left(\frac{\partial\alpha_j^{(e)}}{\partial x}\frac{\partial\alpha_i^{(e)}}{\partial x}+\frac{\partial\alpha_j^{(e)}}{\partial y}\frac{\partial\alpha_i^{(e)}}{\partial y}\right)d\Omega+\iint_{\Omega_e} j\omega\sigma\,\alpha_j^{(e)}\alpha_i^{(e)}\,d\Omega\right] \qquad (V.24)$$

où Ω_e est le domaine de l'élément fini (e).

En pratique, on calcule les contributions élémentaires de chaque élément $M_{ij}^{(e)}$ qu' on "assemble" pour construire la matrice globale du système qui a la particularité d'être creuse.

2.3 Optimisation et C.A.O.

La méthode éléments finis permet de ramener l'équation aux dérivées partielles (V.11) à la résolution d'un système linéaire :

$$M\{A\}=S \qquad (V.25)$$

où M est une matrice complexe appelée matrice d'assemblage, $\{A\}$ le vecteur des N_n valeurs inconnues de la variable d'état aux noeuds de la discrétisation et S le second membre. En introduisant la partie

réelle de P et la partie imaginaire Q de la matrice M on a:

$$M = P + jQ \tag{V.26}$$

où (en simplifiant les notations) :

$$P = \sum_{elements} P_e = \sum_{elements} \iint_{\Omega_e} \nu \{\nabla\alpha\}\{\nabla\alpha\}^T dxdy \tag{V.27}$$

$$Q = \sum_{elements} Q_e = \sum_{elements} \iint_{\Omega_e} \omega\sigma \{\alpha\}\{\alpha\}^T dxdy \tag{V.28}$$

$$S = \sum_{elements} S_e = \sum_{elements} \iint_{\Omega_e} J_S \{\alpha\} dxdy \tag{V.29}$$

où $\{\alpha\}$ représente l'ensemble des fonctions de forme.

Une fois les termes des matrices (V.27), (V.28), (V.29) calculés, le système est résolu par un algorithme numérique de type ICCG (Incomplete Cholesky Conjugate Gradient), bien adapté aux systèmes creux, et qui fournit les valeurs aux noeuds du potentiel vecteur. Les valeurs de A, pour les autres points du domaine, sont ensuite données par l'équation (V.16). Finalement les grandeurs locales comme l'induction B, le champ H, ou les grandeurs globales comme l'énergie magnétique, qui sont liées au potentiel vecteur, seront calculées par des post-calculs effectués après l'étape de résolution. Ce sont ces grandeurs qui seront utilisées pour l'évaluation de la fonction objectif.

Au cours d'un processus d'optimisation, les paramètres géométriques et physiques du problème seront amenés à évoluer. Il nous faut donc préciser ce sur quoi ces paramètres influent dans la méthode éléments finis. Les paramètres physiques $(\sigma, \omega, \nu, J_S)$ n'interviennent que dans la formulation, tandis que les paramètres géométriques, en changeant de valeur, introduisent des perturbations dans le maillage et donc modifient, à la fois, les domaines élémentaires Ω_e et les fonctions de forme qui leur sont attachées. Par conséquent, lors d'une modification d'un paramètre physique, seule l'étape de résolution est à refaire alors que dans le cas d'un paramètre géométrique, il est nécessaire en plus de réaliser un nouveau maillage en amont (Fig. V.1):

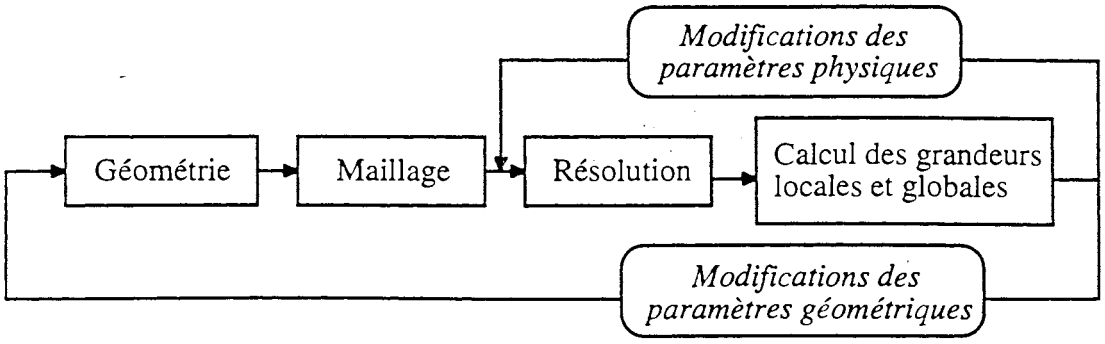


Figure V.1: Influences d'une modification de paramètre sur les calculs éléments finis

Il reste cependant quelques points à préciser avant de pouvoir intégrer une boucle d'optimisation dans un logiciel éléments finis. Notamment, lorsqu'un algorithme de gradient est employé, il est nécessaire de disposer d'un outil de calcul des dérivées premières de la fonction objectif et des contraintes. Sur ce point deux types de méthodes peuvent être mises en œuvre. La première consiste à utiliser une approximation par différences finies [Kadded92]. Cette approche est particulièrement intéressante puisqu'elle est très simple à implanter et qu'elle reste très générale. Cependant, on lui préfère les méthodes qui se basent non pas sur une approximation mais sur le calcul exact de la dérivée de la variable d'état. Cette technique a été développée par [Kouyoumdjian85] et a été utilisée pour l'optimisation de forme par [Kadded92] et [DeVasconcelos94a].

Le maillage est l'autre point fondamental. Au laboratoire d'électrotechnique de Grenoble, le logiciel éléments finis Flux2d utilise, pour cette étape, une triangulation de Delaunay qui garantit la conformité du maillage (cf. ChapIV). En principe, une modification de paramètre demande un remaillage complet (ou partiel) du domaine. Malheureusement cette opération entraîne non seulement une quantité de calcul non négligeable mais, en plus, elle provoque un bruit numérique lors du calcul des gradients [Weeber92]. Pour éviter ce genre de problème, on a souvent recours à des outils de maillage dit *élastiques* qui garantissent une topologie constante lors d'une modification faible de la géométrie. On peut citer, par exemple, le maillage élastique de [Albertini88] ou les travaux de [Weeber92] sur ce thème. Cependant, pour des variations importantes des paramètres, la qualité des éléments triangulaires se dégrade rapidement et il est alors nécessaire d'effectuer un remaillage nécessitant des modifications topologiques.

Ces points étant précisés, on peut dresser l'architecture générale d'un logiciel de CAO d'optimisation de forme (Voir Fig. V.2):

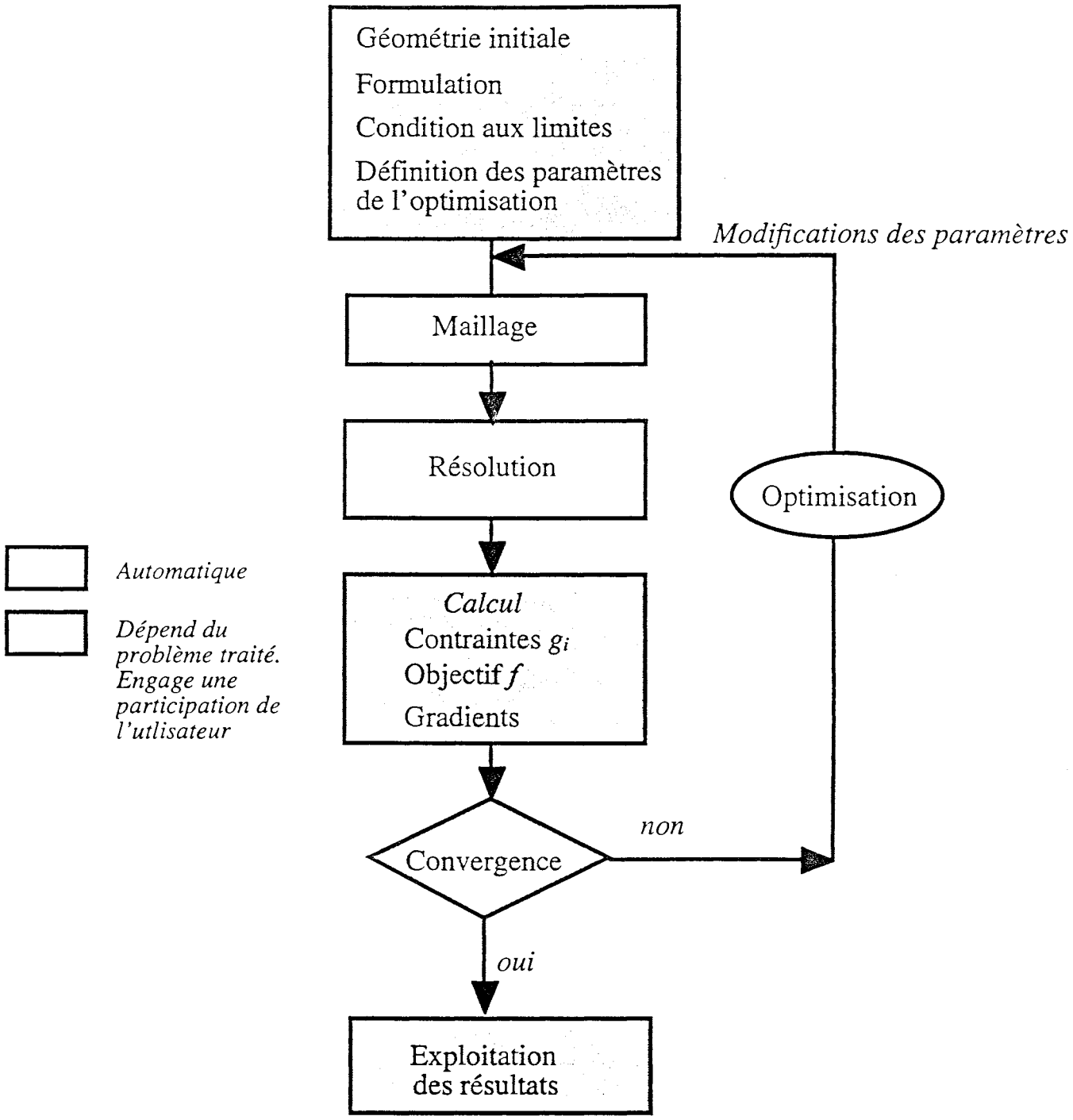


Figure V.2: Architecture d'un logiciel pour l'optimisation de forme

2.4 Pourquoi ne pas s'arrêter là ?

Cette approche pour l'optimisation de forme, quoique très générale, est en fait très contraignante, car si on désire implanter un algorithme génétique dans la boucle d'optimisation, pour tirer profit de sa capacité à localiser l'optimum global, on se heurte alors à un problème de poids: le temps de calcul des évaluations. En effet, même si le temps de remaillage peut être réduit grâce à l'utilisation de techniques

élastiques, il n'en va pas de même pour le temps de résolution, et dans le cas d'une approche génétique, les coûts totaux d'évaluation de l'objectif et des contraintes risquent finalement d'être exorbitants !

Dans la section précédente, nous avons évoqué la nécessité d'un module de calcul des dérivées pour des optimisations de type gradient. Ces calculs sont souvent considérés comme secondaires, pourtant ils peuvent se révéler très utiles si on les regarde d'un autre point de vue: le gradient n'est pas seulement une information indispensable pour l'optimisation d'une fonction f mais c'est aussi une information permettant d'obtenir une approximation (au 1er ordre) de cette fonction. En effet, en notant p la valeur d'un paramètre, on peut prévoir la valeur de f pour la valeur $p+\delta p$ grâce à la formule:

$$f(p+\delta p) \approx f(p)+\delta p f'(p)$$

Ainsi, on peut espérer remplacer les résolutions successives par un calcul de dérivée, pour une valeur initiale de p, et des évaluations (instantanées!) de l'approximation pour les autres valeurs du paramètre. Bien sûr, cette approximation n'est exacte que pour des fonctions linéaires par rapport à p et elle devient très mauvaise dans d'autres situations. Pour améliorer la précision on peut alors calculer les dérivées d'ordre supérieur et finalement utiliser, pour l'approximation de f, un développement en série de Taylor de la fonction:

$$f(p+\delta p) \approx f(p)+\delta p f'(p)+\frac{\delta p^2}{2} f''(p)+\cdots+\frac{\delta p^n}{n!} f^{(n)}(p)$$

où n désigne l'ordre du développement.

Le calcul de toutes les dérivées est évidemment coûteux, mais une fois réalisé, le développement de Taylor permettra des évaluations de la fonction quasi-instantanées. On voit donc tout l'intérêt que l'on peut tirer de l'association de cette méthode d'évaluation avec un algorithme d'optimisation génétique : posséder un outil d'optimisation de forme, à la fois performant (localisation de l'optimum global) et peu coûteux en temps de calcul.

3 CALCUL DES DERIVEES D'ORDRE ELEVE ET CONSTRUCTION DU DEVELOPPEMENT DE TAYLOR

Nous nous plaçons dans le cadre de problèmes bidimensionnels magnétodynamiques linéaires. Nous présenterons les calculs des dérivées d'ordre élevé sur la variable d'état qui est ici le potentiel vecteur A [Coulomb96][Petin96].

3.1 Principe général

La méthode éléments finis conduit à la résolution du système linéaire (V.25). Les termes (V.27), (V.28), (V.29) restent valables quels que soient les valeurs des paramètres géométriques ou physiques (la solution est stationnaire [Coulomb83]). On peut donc dériver par rapport à un paramètre p

quelconque l'équation (V.25), ce qui donne:

$$M\frac{\partial A}{\partial p}=\frac{\partial S}{\partial p}-\frac{\partial M}{\partial p}A \tag{V.30}$$

Par récurrence, on obtient alors les dérivées d'ordre supérieur:

$$M\frac{\partial^m A}{\partial p^m}=\frac{\partial^m S}{\partial p^m}-\sum_{i=1}^{i=m}C_m^i\frac{\partial^i M}{\partial p^i}\frac{\partial^{m-i}A}{\partial p^{m-i}} \tag{V.31}$$

avec $C_m^i=\dfrac{m!}{i!(m-i)!}$ *et* $C_m^0=C_m^m=1.$

Enfin, avec ces dérivées, on peut construire le polynôme de Taylor qui permet d'obtenir la valeur du potentiel vecteur, pour n'importe quelle valeur de p, à partir de la solution en une valeur centrale p_0:

$$A(p_o+\delta p)=\sum_{i=0}^{N_p}\frac{\delta p^i}{i!}\frac{\partial^i A}{\partial p^i}(p_0) \tag{V.32}$$

où N_p désigne l'ordre du développement de Taylor par rapport au paramètre p.
Dans le cas de plusieurs paramètres, on doit non seulement calculer les dérivées précédentes mais en plus les dérivées croisées de A par rapport aux différents paramètres. Par exemple, pour 2 paramètres indépendants p et q, on obtient une généralisation de (V.31) et (V.32):

$$M\frac{\partial^{m+n}A}{\partial p^m\partial q^n}=\frac{\partial^{m+n}S}{\partial p^m\partial q^n}-\sum_{\substack{i=0\\i+j>0}}^{m}\sum_{j=0}^{n}C_m^iC_n^j\frac{\partial^{i+j}M}{\partial p^i\partial q^j}\frac{\partial^{m-i+n-j}A}{\partial p^{m-i}\partial q^{n-j}} \tag{V.33}$$

$$A(p_o+\delta p,q_0+\delta q)=\sum_{i=0}^{N_p}\sum_{j=0}^{N_q}\frac{\delta p^i}{i!}\frac{\delta q^j}{j!}\frac{\partial^{i+j}A}{\partial p^i\partial q^j}(p_0,q_0) \tag{V.34}$$

3.2 Quelques remarques

- Le calcul des dérivées successives s'obtient par la résolution du système (V.33) qui possède la même matrice que le système éléments finis initial et dans lequel seul le second membre est différent.

- Les trois termes élémentaires (V.27) (V.28) (V.29) sont linéaires par rapport aux paramètres physiques $\omega\sigma, \nu, J_S$. Ceci implique que les dérivées d'ordre 1 par rapport à ces paramètres sont constantes et que les dérivées d'ordre supérieur sont nulles. Cependant, ce type de dépendance, bien que linéaire n'entraîne pas nécessairement une linéarité de la variable d'état par rapport à ces paramètres. En effet, les paramètres $\omega\sigma, \nu$ influent directement sur la matrice d'assemblage M.

- Le problème, bien que linéaire, peut présenter des sous-domaines où les paramètres physiques ont des valeurs constantes mais différentes. Dans la suite, on notera D_k ces sous-domaines auxquels seront associés les paramètres physiques ($\omega\sigma_k, \nu_k, J_{Sk}$).

- Comme nous l'avions souligné plus haut, les paramètres géométriques influent sur les termes (V.27) (V.28) (V.29) car ils modifient les coordonnées des noeuds de la discrétisation et donc les domaines élémentaires Ω_e. De plus, cette dépendance n'est pas a priori linéaire et il existera donc en général des dérivées d'ordre élevé pour les paramètres géométriques.

Nous allons maintenant détailler les calculs des dérivées de M et de S indispensables pour obtenir le développement de Taylor de A.

3.3 Dérivations de M et S par rapport aux paramètres physiques

3.3.1 Paramètre ν_k

A l'aide des équations (V.26) et (V.27) on déduit :

$$\begin{cases} \dfrac{\partial M}{\partial \nu_k} = \displaystyle\sum_{\Omega_e \in D_k} \iint_{\Omega_e} \{\nabla\alpha\}\{\nabla\alpha\}^T dxdy \\ \dfrac{\partial^n M}{\partial \nu_k^n} = 0 \quad pour\ n \geq 2 \end{cases} \tag{V.35}$$

$$\frac{\partial^n S}{\partial \nu_k^n} = 0 \quad pour\ n \geq 1 \tag{V.36}$$

3.3.2 Paramètre $\omega\sigma_k$

Des équations (V.26) et (V.28) on déduit:

$$\begin{cases} \dfrac{\partial M}{\partial(\omega\sigma_k)} = j \displaystyle\sum_{\Omega_e \in D_k} \iint_{\Omega_e} \{\alpha\}\{\alpha\}^T dxdy \\ \dfrac{\partial^n M}{\partial(\omega\sigma_k)^n} = 0 \quad pour\ n \geq 2 \end{cases} \tag{V.37}$$

$$\frac{\partial^n S}{\partial(\omega\sigma_k)^n} = 0 \quad pour\ n \geq 1 \tag{V.38}$$

3.3.3 Paramètre J_{Sk}

$$\frac{\partial^n M}{\partial J_{Sk}^n} = 0 \quad pour\ n \geq 1 \tag{V.39}$$

$$\begin{cases} \dfrac{\partial S}{\partial J_{Sk}} = \displaystyle\sum_{\Omega_e \in D_k} \iint_{\Omega_e} \{\alpha\} dxdy \\ \dfrac{\partial^n S}{\partial J_{Sk}^n} = 0 \quad pour\ n \geq 2 \end{cases} \tag{V.40}$$

Les sommations ci-dessus sont faites uniquement sur les éléments appartenant aux sous-domaines D_k. Comme nous l'avons mentionné précédemment, les dérivées d'ordre supérieur à un sont nulles et par conséquent les dérivées croisées entre ces différents paramètres physiques aussi.

3.4 Dérivation de M et S par rapport aux paramètres géométriques

Avant de poursuivre les calculs il nous faut introduire ce qu'est l'*élément de référence*. Cet élément n'est pas spécifique à la méthode de dérivation mais est utilisé classiquement dans la méthode des éléments finis pour alléger les notations lors du calcul des intégrales.

Les intégrales qui apparaissent dans les équations (V.27) à (V.29) sont le plus souvent calculées par une quadrature numérique comme par exemple celle de Gauss [Touzot84]. Afin d'éviter le calcul des coordonnées des points de Gauss pour chaque élément fini, on introduit un élément que l'on nomme élément de référence, défini dans un repère local *(u,v)* et pour lequel on effectue ces calculs. Les éléments du maillage peuvent être vus comme l'image de l'élément de référence par une transformation adéquate. Ainsi, le calcul d'une intégrale sur un élément fini quelconque se ramène au calcul d'une intégrale sur l'élément de référence grâce à un changement de variables (Fig. V.3).

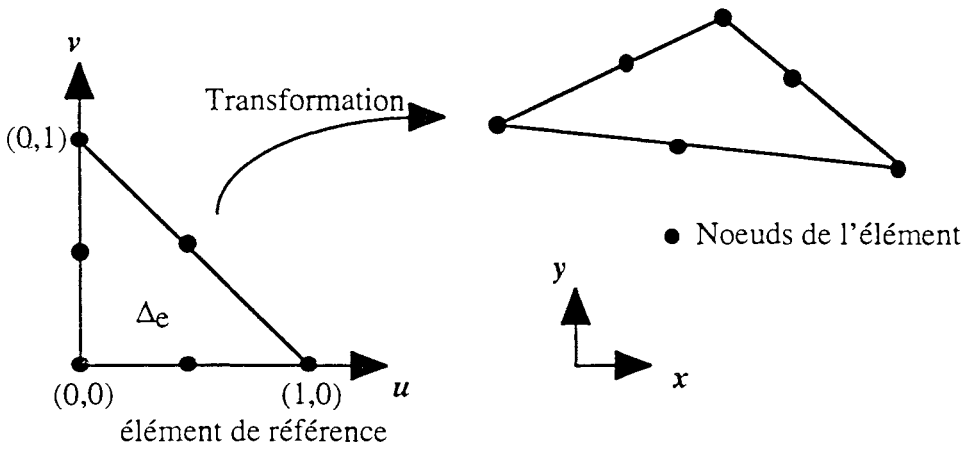


Figure V.3: Elément de référence

Par exemple, si on considère le second membre S défini par (V.29) alors en utilisant le repère local on se ramène à:

$$S = \sum_{éléments} \iint_{\Delta_e} J_S\{\alpha\}|G|dudv \tag{V.41}$$

où Δ_e désigne l'élément de référence et $|G|$ le déterminant de la matrice Jacobienne de la transformation du repère local *(u,v)* vers le repère *(x,y)* par rapport auquel sont définis tous les éléments.

Lorsqu'on modifie un paramètre géométrique *p*, on peut conserver une topologie constante pour le maillage en utilisant des techniques élastiques qui modifient uniquement les coordonnées des noeuds dans le maillage. Ceci entraîne des déformations sur les éléments: certains s'étirent et d'autres s'aplatissent mais si les déformations ne sont pas trop importantes on peut espérer conserver des éléments d'assez bonne qualité. En dérivant (V.41) par rapport à un paramètre géométrique *p*, on a alors:

$$\frac{\partial S}{\partial p} = \sum_{\substack{éléments \\ déformés}} \iint_{\Delta_e} J_S\{\alpha\}\frac{\partial|G|}{\partial p}dudv \tag{V.42}$$

et plus généralement pour des dérivées d'ordre supérieur à 1 on obtient (pour deux paramètres *p* et *q* par exemple):

$$\frac{\partial^{m+n} S}{\partial p^m \partial q^n} = \sum_{\substack{éléments \\ déformés}} \iint_{\Delta_e} J_S\{\alpha\}\frac{\partial^{m+n}|G|}{\partial p^m \partial q^n}dudv \tag{V.43}$$

On peut remarquer que ces intégrales sont limitées aux seuls éléments déformés, ce qui réduit considérablement le coût de calcul.

Pour la matrice Q, la dérivation s'effectue de la même manière et on obtient :

$$\frac{\partial^{m+n} Q}{\partial p^m \partial q^n} = j\sum_{\substack{éléments \\ déformés}} \iint_{\Delta_e} \omega\sigma\{\alpha\}\{\alpha\}^T \frac{\partial^{m+n}|G|}{\partial p^m \partial q^n}dudv \tag{V.44}$$

Le cas de la matrice P est un peu plus compliqué puisque ce sont les gradients des fonctions de forme qui interviennent. Pour chaque élément il faut donc expliciter le vecteur $\{\nabla\alpha\}$ dans le repère *(u,v)*. En notant $\nabla\alpha_i$ le gradient de la fonction de forme *i* de l'élément on a l'expression suivante:

$$\nabla\alpha_i = \begin{bmatrix} \frac{\partial\alpha_i}{\partial x} \\ \frac{\partial\alpha_i}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial v}{\partial x} \\ \frac{\partial u}{\partial y} & \frac{\partial v}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial\alpha_i}{\partial u} \\ \frac{\partial\alpha_i}{\partial v} \end{bmatrix} = G^{-1}\nabla_u\alpha_i \tag{V.45}$$

où G^{-1} est l'inverse de la matrice jacobienne et $\nabla_u \alpha_i$ le gradient (dans le repère local (u,v)) de la fonction de forme i de l'élément de référence.
Finalement le produit scalaire entre fonctions de forme peut s'écrire :

$$\{\nabla\alpha\}\{\nabla\alpha\}^T = \{\nabla_u\alpha\}G^{-1T}G^{-1}\{\nabla_u\alpha\}^T \tag{V.46}$$

En écrivant (V.28) dans le repère (u,v):

$$P = \sum_{éléments} \iint_{\Delta_e} \nu\{\nabla_u\alpha\}|G|G^{-1T}G^{-1}\{\nabla_u\alpha\}^T dudv \tag{V.47}$$

Le seul terme qui dépend de la géométrie est le produit $|G|G^{-1T}G^{-1}$. En dérivant ce produit dans l'expression précédente on obtient les dérivées d'ordre supérieur:

$$\frac{\partial^{m+n}P}{\partial p^m \partial q^n} = \sum_{\substack{éléments \\ déformés}} \iint_{\Delta_e} \nu\{\nabla_u\alpha\}\frac{\partial^{m+n}}{\partial p^m \partial q^n}\left[|G|G^{-1T}G^{-1}\right]\{\nabla_u\alpha\}^T dudv \tag{V.48}$$

Les expressions (V.43) (V.44) et (V.48) dépendent toutes des dérivées du jacobien G, de son déterminant et de son inverse. Cependant, on peut noter que les dérivées de G^{-1} peuvent être simplement déduites de celles de G à l'aide de l'équation $G^{-1}G=Id$. Par exemple, la dérivée première de G^{-1} est :

$$\frac{\partial G^{-1}}{\partial p} = -G^{-1}\frac{\partial G}{\partial p}G^{-1} \tag{V.49}$$

Concernant les termes croisés entre paramètres géométriques et paramètres physiques, ils se limitent à l'ordre 1 pour ces derniers et ils peuvent être déduits des équations précédentes :

$$\frac{\partial^{1+m+n}M}{\partial\nu_k \partial p^m \partial q^n} = \sum_{\substack{éléments \\ déformés \in D_k}} \iint_{\Delta_e} \{\nabla_u\alpha\}\frac{\partial^{m+n}}{\partial p^m \partial q^n}\left[|G|G^{-1T}G^{-1}\right]\{\nabla_u\alpha\}^T dudv \tag{V.50}$$

$$\frac{\partial^{1+m+n}M}{\partial(\omega\sigma_k)\partial p^m \partial q^n} = j\sum_{\substack{éléments \\ déformés \in D_k}} \iint_{\Delta_e} \{\alpha\}\{\alpha\}^T \frac{\partial^{m+n}|G|}{\partial p^m \partial q^n} dudv \tag{V.51}$$

$$\frac{\partial^{1+m+n}S}{\partial J_{Sk} \partial p^m \partial q^n} = j\sum_{\substack{éléments \\ déformés \in D_k}} \iint_{\Delta_e} \{\alpha\}\frac{\partial^{m+n}|G|}{\partial p^m \partial q^n} dudv \tag{V.52}$$

4 UNE NOUVELLE APPROCHE POUR L'OPTIMISATION DE FORME

En pratique, pour construire le développement de Taylor, on calcule, à l'aide des formules (V.35) à (V.52), les dérivées d'ordre élevé de M et S pour des valeurs, dites centrales, des paramètres (notées p_0, q_0, ...). Ensuite, les dérivées de la variable d'état A sont calculées de façon récursive en résolvant le système (V.31). Comme nous l'avions mentionné, les systèmes (V.31) possèdent la même matrice M et seuls les seconds membres sont différents. Il est donc préférable d'employer une méthode directe pour les résoudre plutôt qu'une méthode itérative de type gradient conjugué. En effet, dans ce cas, seule la première résolution demandera un effort important de calcul; les résolutions suivantes seront alors obtenues à partir de la première par des substitutions peu coûteuses.

Finalement, pour toutes les variations δp, δq des paramètres par rapport à leur valeur centrale, on peut évaluer la nouvelle valeur de A à l'aide de (V.34) et ce, de façon instantanée.
La Figure V.4 montre comment introduire ces nouveaux développements en vue de l'optimisation de forme. Le polynôme étant construit sur le potentiel vecteur, il est nécessaire d'effectuer des post-traitements pour calculer les grandeurs locales et globales qui interviennent dans l'objectif et les contraintes.

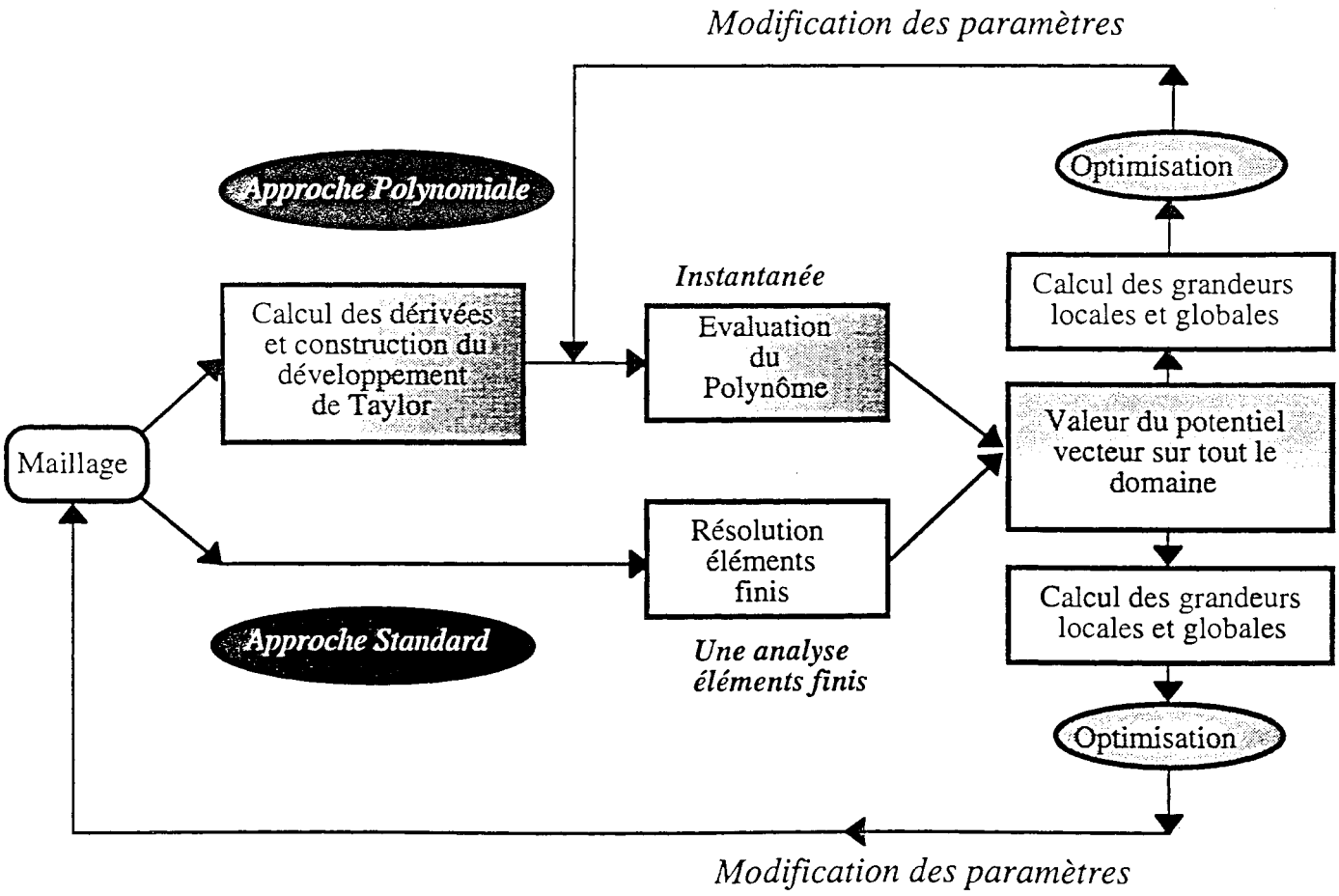


Figure V.4: Comparaison des étapes entre l'approche polynomiale et l'approche éléments finis standard

Du point de vue logiciel, nous avons utilisé une "boîte noire" pour le calcul du développement de Taylor. Cette "boîte" est la propriété de la société CADOE et est intégrée dans le logiciel [FLUX-PARAM] qui réalise les post-traitements. Nous avons donc couplé nos algorithmes d'optimisation avec FLUX-PARAM pour constituer un logiciel pour l'optimisation de forme.

Il reste cependant deux points que nous n'avons pas abordés et qui sont fondamentaux: la **précision du développement** et le **rayon de convergence de la série**.

La précision du développement correspond aux nombres de termes (N_p) qu'il faut calculer dans le développement pour obtenir une précision donnée sur la variable A. Cette notion ne doit pas être confondue avec le rayon de convergence qui se réfère à la variation maximum δp_{max} du paramètre p pour lequel le développement converge effectivement vers A. Pour fixer les idées on peut prendre par exemple les fonctions exponentielle $exp(x)$ et $\frac{1}{1-x}$.

- $exp(x)$ possède un rayon de convergence infini c'est-à-dire que: $\forall x \in \Re \quad exp(x) = \sum \frac{x^n}{n!}$. De plus, la précision est d'autant meilleure que le nombre de termes est important.

- $\frac{1}{1-x}$ possède un rayon de convergence égal à 1 ce qui veut dire que: $\forall x \in [-1,1] \quad \frac{1}{1-x} = \sum x^n$. En revanche, en dehors de cet intervalle, l'égalité n'est plus vraie et augmenter le nombre de termes significatifs dans le développement ne changera rien à la divergence de la série !

Le fait que les calculs des dérivées soient réalisés par le logiciel de CADOE ne nous permet pas (confidentialité oblige) d'avoir accès à ces informations essentielles sur les problèmes que l'on désire étudier. On trouve cependant des résultats intéressants sur ce délicat problème dans les travaux de [Petin96] et [Guillaume94].

Pour le paramètre J_s le rayon de convergence est infini puisque c'est un terme source, par contre pour les autres paramètres physiques ν et $\omega\sigma$ qui interviennent dans la matrice d'assemblage, les rayons de convergence sont respectivement $]0, 2\nu_0[$ et $]0, 2(\omega\sigma)_0[$ où ν_0 et $(\omega\sigma)_0$ désignent les valeurs centrales de ces paramètres [Coulomb96]. Cependant, pour obtenir une bonne précision, avec un ordre N_p raisonnable (10 à 20 termes), les domaines d'utilisation doivent être restreints aux intervalles $]0.2, 1.8\nu_0[$ et $]0.2, 1.8(\omega\sigma)_0[$ [Petin96].

Concernant les paramètres géométriques, il n'existe pas de résultats généraux au sujet de la convergence du développement de Taylor. Les tests réalisés par [Guillaume94] semblent indiquer que des développements à des ordres compris entre 5 et 10 pour la géométrie donnent une très bonne précision (quelques %). L'autre point concerne le rayon de convergence. Sur les problèmes qu'il a étudié, [Guillaume94] met en évidence l'influence du maillage sur le rayon de convergence de la série.

Notamment il note que pour un maillage initial de "qualité", il n'est pas nécessaire d'avoir un maillage déformé d'aussi bonne qualité car, même si la série diverge, les résultats obtenus donnent une bonne précision sur la valeur de A. Mais d'une façon générale, nous n'avons pas trouvé de méthode effective pour déterminer le rayon de convergence de la série, ni même pour déterminer l'ordre du développement nécessaire pour obtenir une bonne approximation de la variable A.

Les principes généraux de notre approche étant décrits, il nous reste à traiter un problème test "réel" pour montrer les avantages d'un couplage d'une méthode génétique avec une méthode d'approximation par développement de Taylor.

5 PROBLEME TEST

5.1 Généralités sur le SMES

Les SMES (Superconducting Magnetic Energy Storage) sont des dispositifs qui stockent de grandes quantités d'énergie sous forme magnétique. Ils sont constitués de matériaux supraconducteurs qui permettent d'avoir une résistance électrique nulle moyennant de très basses températures. Schématiquement, le SMES peut être vu comme une bobine supraconductrice attachée à un dispositif qui charge et décharge la bobine (Fig. V.5).

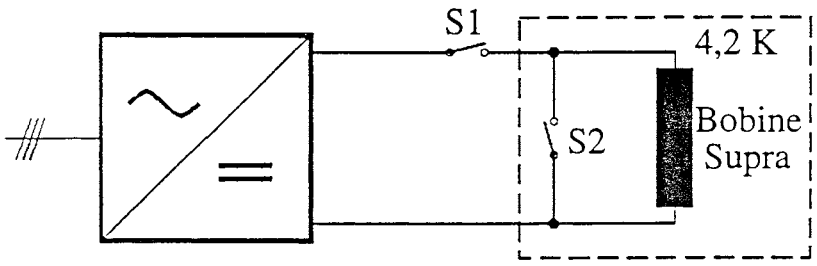


Figure V.5: Schéma de principe d'un SMES
(Les fonctions de S1 et S2 sont assurées par le convertisseur)

La bobine se charge lorsque l'interrupteur S1 est fermé. Une fois la bobine chargée, on ferme S2 puis on ouvre l'interrupteur S1 : le courant circule alors dans la bobine sans aucune perte puisque le SMES est à l'état supraconducteur. Le champ magnétique créé alors par le courant permet de conserver directement de l'énergie sous forme magnétique sans aucune conversion. Dans les années 80, de nombreux projets ont vu le jour, impliquant l'utilisation de SMES constitués de bobines de plusieurs centaines de mètres de diamètre (Fig. V.6) mais aujourd'hui les recherches sont plutôt axées sur des SMES de dimensions "raisonnables".

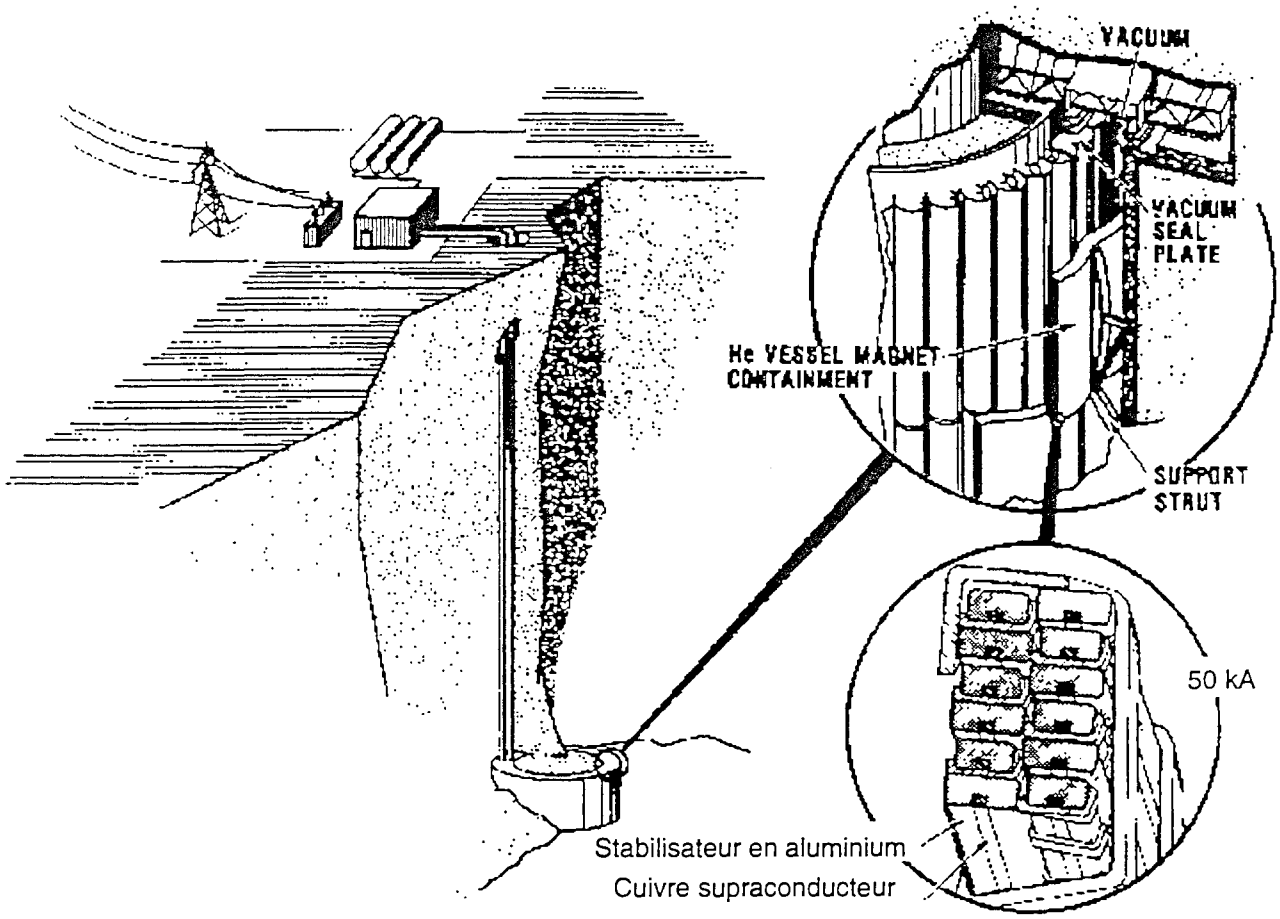


Figure V.6: Vision d'artiste pour un projet de SMES de 5000 Mwh

Deux types de bobines peuvent être utilisées dans les SMES: le solénoïde et le tore (Fig. V.7). L'intérêt d'une bobine toroïdale est que le champ magnétique en dehors du dispositif est nul et même si ceci n'est valable que dans le cas d'un enroulement parfait, il semble que cela soit une solution plus prometteuse que le solénoïde. Une autre solution pour diminuer le champ à l'extérieur du dispositif consiste à utiliser deux solénoïdes parcourus par des courants de sens opposés.

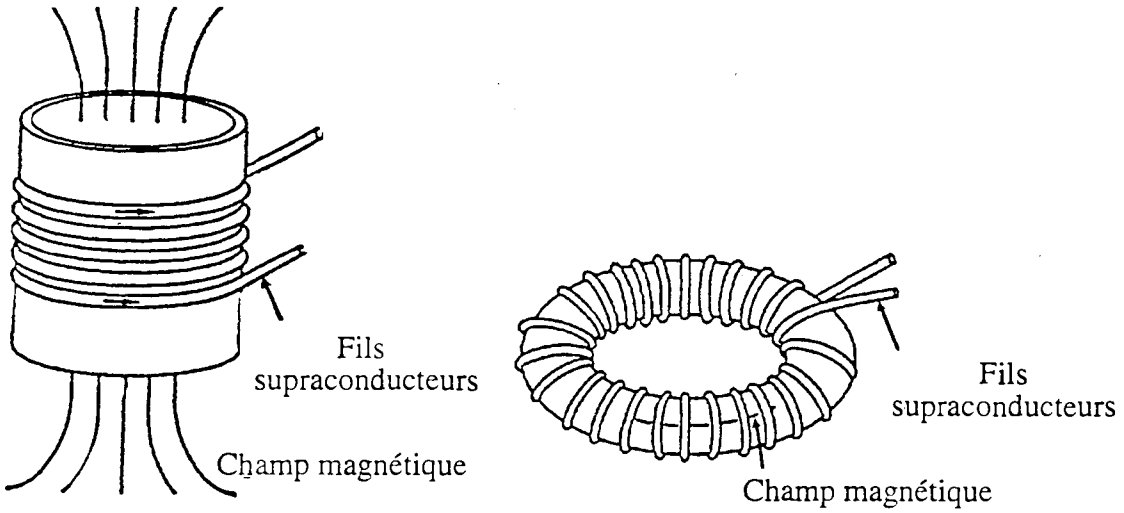


Figure V.7: Bobines solénoïdale et toroïdale pour le SMES

Cette dernière solution a été retenu dans le problème TEAM workshop *problem 22* que nous allons traiter.

5.2 Optimisation

Le problème TEAM workshop n°22 [TEAM22] est constitué de deux bobines supraconductrices et possède 8 degrés de liberté (Fig. V.8). L'optimisation consiste à trouver les "bonnes valeurs" de paramètres géométriques (R_1, R_2, h_1, h_2, d_1, d_2) et physiques (J_1, J_2) de façon à satisfaire les deux objectifs suivants:

- L'énergie stockée sous forme magnétique dans le dispositif doit être égale à 180MJ
- L'induction, le long de deux lignes situées à 10 mètres du dispositif, doit être la plus faible possible.

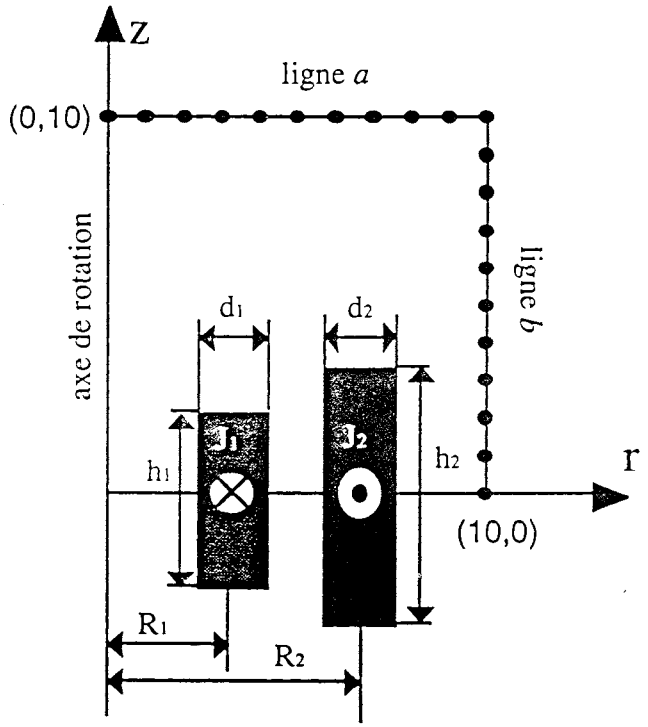


Figure V.8: Paramètres du SMES

Les domaines de variations des paramètres sont présentés dans la table V.1. A ces contraintes, il faut ajouter une condition qui assure que les deux bobines ne se chevaucheront pas. Une manière simple, pour éviter d'introduire de nouvelles équations tout en satisfaisant cette condition, consiste à définir deux nouveaux paramètres R_3 et R_4 à la place de R_1 et R_2 comme présenté sur la figure V.9. Les domaines de variations pour ces paramètres sont montrés dans la table V.2.

	R_1 [mm]	R_2 [mm]	$h_1/2$ [mm]	$h_2/2$ [mm]	d_1 [mm]	d_2 [mm]	J_1 [A/mm²]	J_2 [A/mm²]
min.	1000	1800	100	100	100	100	10	-30
max.	4000	5000	1800	1800	800	800	30	-10

Table V.1: Contraintes de domaines sur les paramètres du SMES

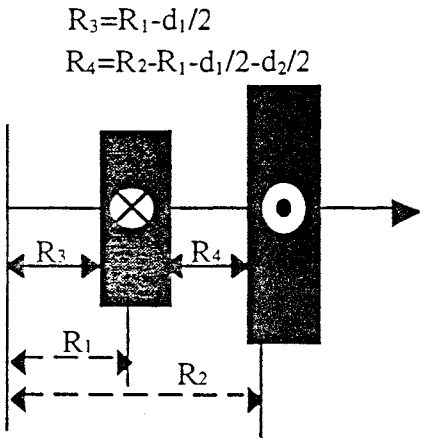


Figure V.9: Définition de nouveaux paramètres pour le SMES

	R_3 [mm]	R_4 [mm]
min.	600	0
max.	3950	3900

Table V.2: Contraintes de domaines sur les nouveaux paramètres R_3 et R_4

Pour conserver le caractère supraconducteur des bobines il est nécessaire que les grandeurs densité de courant, température et induction soient inférieures à certaines valeurs critiques. Celles-ci sont liées entre elles et forment une surface critique dans l'espace induction, température et densité de courant. Ainsi, dans le processus d'optimisation cette condition ("quench condition" en anglais) se traduira, pour chaque bobine, par une contrainte sur la densité de courant et la valeur maximum de l'induction à l'intérieur de la bobine. Cette courbe "critique" est en général approximée par une droite au dessus de laquelle le matériau perd ses propriétés supraconductrices (Fig. V.10).

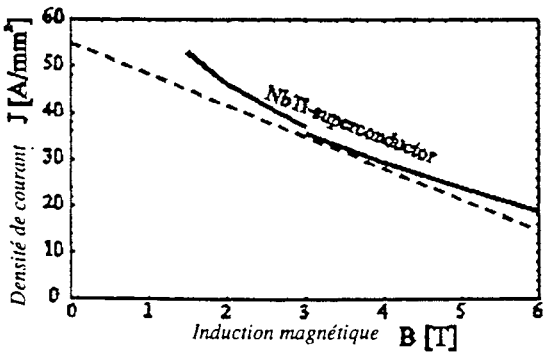


Figure V.10: Courbe critique d'un supraconducteur industriel

5.3 Formulation mathématique du problème

Les deux objectifs sont couplés au moyen d'une somme pondérée de deux termes normalisés:

$$f = w_1 \frac{B_{stray}^2}{B_{nom}^2} + w_2 \frac{|Energie - E_{ref}|}{E_{ref}} \tag{V.53}$$

où: $B_{norm} = 2.0E-4\ T$, $E_{ref} = 180\ MJ$ et

$$B_{stray}^2 = \frac{\sum_{i=1}^{i=22} |B_{stray_i}|^2}{22} \tag{V.54}$$

B_{stray_i} désigne l'induction évaluée en 22 points répartis uniformément sur les lignes a et b (Fig. V.8).
Les contraintes physiques sur les bobines peuvent être approximées par les deux inégalités:

$$\begin{aligned} J_1 &\leq (-6.4|B_{max_1}| + 54.)\ A/mm^2 \\ J_2 &\leq (-6.4|B_{max_2}| + 54.)\ A/mm^2 \end{aligned} \tag{V.55}$$

Où B_{max_i} désigne la valeur maximum de l'induction dans la bobine i.
Ces deux contraintes seront associées à l'objectif sous la forme d'une fonction de pénalisation.

Ce problème est particulièrement intéressant dans le cadre de l'optimisation stochastique puisqu'il possède de nombreux optima locaux [Alotto95].

6 RESULTATS

6.1 Le Logiciel Flux-param - Développements de Taylor obtenus

Le logiciel FLUX-PARAM est composé d'un certain nombre de modules (Fig. V.11):

prepar : Ce module sert à la préparation de la géométrie et du maillage initial. L'utilisateur y rentre les valeurs centrales des paramètres géométriques utilisées pour la construction du développement de Taylor ainsi que les plages de variations qu'il désire sur ces paramètres.

pargeo : Ce module prépare les paramètres géométriques pour la résolution. Il utilise les programmes FLUXMESH et ALGOMESH développés par la société CADOE. Ce programme vérifie que la géométrie reste cohérente et que le maillage n'est pas trop perturbé lorsque les différents paramètres varient. S'il détecte des anomalies alors le programme s'arrête et l'utilisateur est chargé de modifier ses plages de variations des paramètres (et donc de retourner dans le module **prepar**).

Malheureusement, nous ne disposons d'aucune information pertinente sur les modifications les plus adéquates à effectuer en cas d'arrêt de ce module.

propar : Dans ce module, l'utilisateur affecte les propriétés physiques et les conditions aux limites de son problème et ensuite il fixe les valeurs centrales des paramètres physiques ainsi que les plages de variations qu'il désire sur ces paramètres.

respar : Résolution paramétrée qui construit le développement de Taylor.

exppar : Module d'exploitation des résultats: calculs des grandeurs locales et globales à partir du potentiel vecteur (obtenu par le développement de Taylor). C'est dans ce module que nous avons intégré notre algorithme d'optimisation.

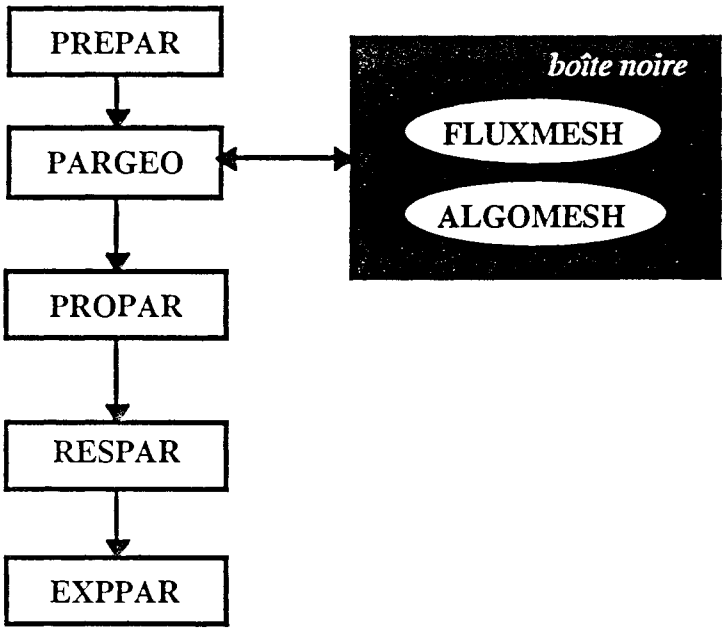


Figure V.11: Architecture du logiciel FLUX-PARAM

Nous nous sommes heurtés à une première limitation technique sur le nombre de paramètres géométriques. En effet, le logiciel a refusé de développer plus de 4 termes géométriques mais il semble que ceci soit dû à la "version 0.0" du logiciel installée au Laboratoire. Par conséquent, toutes nos expériences ont été réalisées avec quatre paramètres géométriques mais cette limitation purement technique n'enlève rien à la généralité de l'approche polynomiale.

Dans une première expérience, nous avons fixé les paramètres R_4 et h_1 (Table V.3) et nous avons réalisé un maillage "grossier" du domaine (Fig. V.12). Le développement a été construit avec les plages de variation que nous avions demandées (Table V.3).

	R_3 [mm]	R_4 [mm]	$h_1/2$ [mm]	$h_2/2$ [mm]	d_1 [mm]	d_2 [mm]	J_1 [A/mm²]	J_2 [A/mm²]
min.	600	-	-	100	100	100	10	-30
max.	3950	-	-	1800	800	800	30	-10
Taylor	**2100**	**1000**	**780**	**1050**	**450**	**450**	**20**	**-20**

Table V.3: Valeurs centrales, minimums et maximums des paramètres 1er test

Les ordres de développement des paramètres sont montrés dans la table V.4. On trouve des ordres compris entre 5 et 7 pour les paramètres géométriques, ce qu'avait constaté [Guillaume94] et de 1 pour les termes sources.

	R_3	h_2	d_1	d_2	J_1	J_2
ordre	6	7	5	5	1	1

Table V.4: Ordre de dérivation des paramètres sur le 1er test

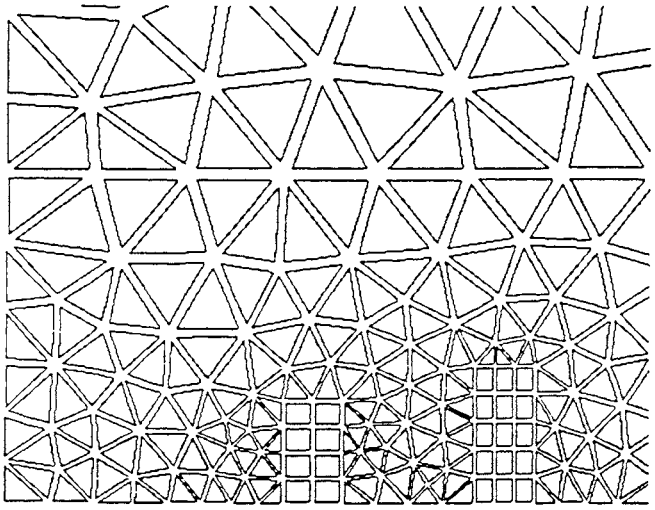


Figure V.12: Vue partielle du maillage initial réalisé sur le 1er test

Nous avons voulu connaître l'influence du maillage initial sur les plages de variation des paramètres. Nous avons donc réalisé des maillages plus fins du domaine. Nous avons constaté que le module **pargeo** refusait des plages de variation aussi importantes que pour notre premier maillage. Il semble donc que les déformations d'éléments soient plus facilement contrôlables lorsque les mailles sont plus "grossières".

Enfin, nous avons voulu tester notre problème en fixant les paramètres h_1 et d_2 aux valeurs optimales présentées dans le problème 22 (Table V.5), dans l'espoir que notre algorithme génétique localise le même optimum. Les plages de variation des autres paramètres sont présentées dans la table V.5 et on constate que nous n'avons pas pu développer le paramètre R_4 sur une plage aussi large que celle

désirée (Table V.2). Il semble en effet, très difficile aux éléments de supporter des déformations engendrées par des variations de ce paramètre. On peut voir sur la figure V.13 le maillage initial et un maillage déformé. On constate que de grandes déformations entraînent rapidement une dégradation de la qualité des éléments, ce qui explique que le module pargeo refuse des plages de variation plus importantes sur ce problème.

	R_3 [mm]	R_4 [mm]	$h_1/2$ [mm]	$h_2/2$ [mm]	d_1 [mm]	d_2 [mm]	J_1 [A/mm²]	J_2 [A/mm²]
min.	600	100	-	400	100	-	10	-30
max.	3950	650	-	1800	800	-	30	-10
Taylor	**2100**	**400**	**784.6**	**1050**	**450**	**256.2**	**20**	**-20**

Table V.5: Valeurs centrales, minimums et maximums des paramètres 2ème test

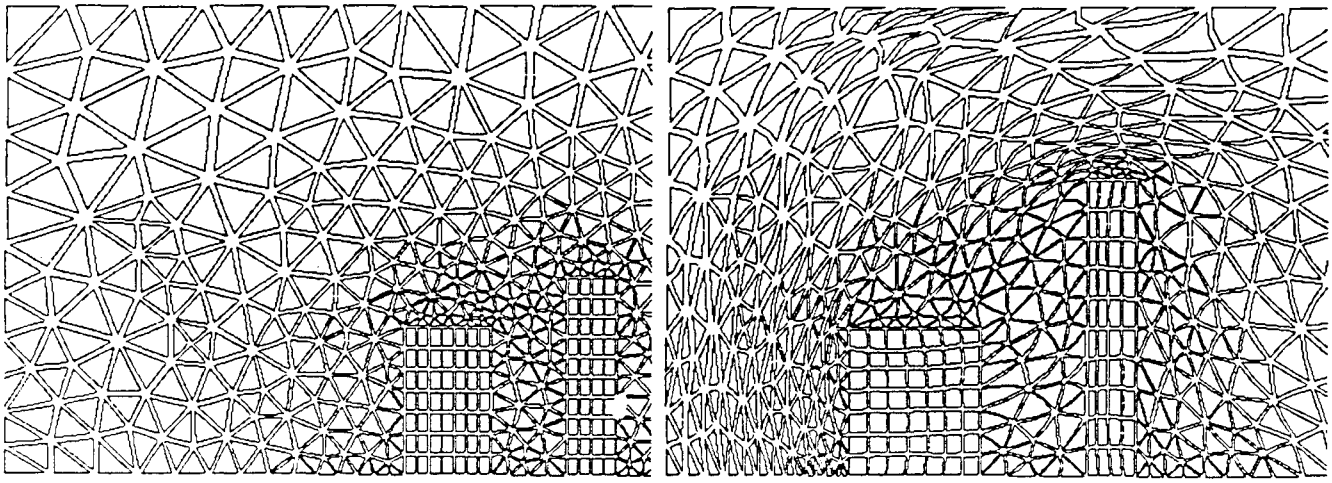


Figure V.13: Maillage initial et Maillage déformé (R_3=700., R_4=600., h_2=1700., d_1=750.)

Les ordres du développement sont présentés dans la table V.6, on retrouve des ordres de 5 à 7 pour les paramètres géométriques.

	R_3	R_4	h_2	d_1	J_1	J_2
ordre	7	5	7	5	1	1

Table V.6: Ordre de dérivation des paramètres sur le 2ème test

6.2 Validité du développement

Pour valider le développement de Taylor, nous avons effectué un certain nombre de ré-analyses par Flux2d pour des valeurs des paramètres éloignées du point central. Les tests comparatifs ont été réalisés sur des grandeurs globales comme l'énergie dans le domaine, ou les lignes d'induction ("équi-flux"). On constate sur la figure V.14, que les "équi-flux" sont sensiblement identiques pour des

valeurs de paramètres très éloignées du point central.

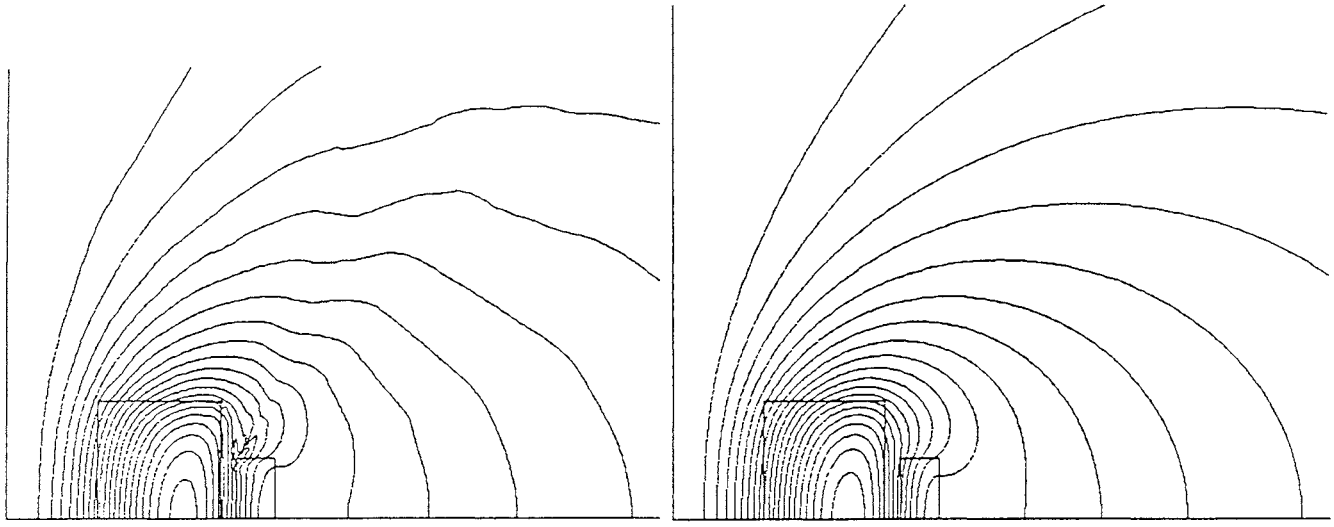


Figure V.14: Comparaison des équi-flux obtenues par le développement et une analyse Flux2d sur le point extrême (R_3=600, R_4=100, $h_2/2$=400, d_1=800)

En ce qui concerne l'énergie, nous l'avons calculée pour des points extrêmes du domaine. Les différences entre la solution obtenue par l'approche polynomiale et celle obtenue par l'approche éléments finis standard sont de l'ordre de 6%. On peut donc considérer que le développement possède une bonne précision sur l'ensemble du domaine.

6.3 Optimisation - Temps de calcul

Pour l'optimisation, nous avons utilisé l'algorithme génétique - décrit dans le chapitre II - pour le deuxième problème. Les poids sur les deux objectifs ont été pris égaux à 0.1 pour l'induction à 10 mètres du dispositif et à 0.9 pour l'énergie. Le réglage des poids s'est avéré assez problématique dans la recherche de l'optimum et on pourra voir une approche basée sur le fuzzy-control pour calculer des poids optimums dans l'article de [Magele96b].

Dans les tables qui suivent nous présentons les valeurs des paramètres à l'optimum, le nombre d'évaluations nécessaire à l'algorithme génétique ainsi que les temps de calcul entre l'approche éléments finis standard et l'approche polynomiale. Le temps de résolution dans l'approche éléments finis standard est calculé comme le produit du temps d'une résolution par le nombre d'évaluations.

On constate que l'algorithme génétique localise bien l'optimum et que le nombre moyen d'évaluations est proche de 10 000 (Table V.7 et V.8). Pour justifier l'utilisation d'un algorithme d'optimisation global, nous avons lancé un algorithme de **Lagrangien Augmenté** à partir d'un point initial quelconque du domaine. Les dérivées premières et secondes sont dans ce cas, calculées par différences finies, ce qui peut paraître contre nature puisque [FLUX-PARAM] a calculé des dérivées d'ordre supérieur pour construire le développement, mais malheureusement, nous n'avons pas accès directement à ces informations. A partir de ce point initial, l'algorithme de Lagrangien Augmenté

converge vers un nouveau point du domaine qui n'est malheureusement pas l'optimum global mais seulement un optimum local (Tables V.9 et V.10). Ceci justifie donc l'emploi d'une méthode stochastique comme l'algorithme génétique que nous avons utilisé.

	R_3 [mm]	R_4 [mm]	$h_2/2$ [mm]	d_1 [mm]	J_1 [A/mm²]	J_2 [A/mm²]
Optimum	**1470.81**	**250.84**	**1307.80**	**609.94**	**14.4577**	**-10.9605**

Table V.7: Valeurs optimales des paramètres obtenues par optimisation génétique

	neval	B*stray*^{2} [T^2]	Energie [MJ]
Objectif	9350	4.9 E-9	179.99

Table V.8: Valeurs des objectifs à l'optimum et nombre d'évaluations nécessaire pour l'AG

	R_3 [mm]	R_4 [mm]	$h_2/2$ [mm]	d_1 [mm]	J_1 [A/mm²]	J_2 [A/mm²]
Point initial	1200	500	1100	450	20	-20
Optimum	**1018.94**	**478.96**	**959.16**	**508.88**	**16.8581**	**-10.006**

Table V.9: Valeurs optimales des paramètres obtenues par Lagrangien Augmenté

	neval	B*stray*^{2} [T^2]	Energie [MJ]
Objectif	255	4.5 E-8	129.21

Table V.10: Valeurs des objectifs à l'optimum et nombre d'évaluations nécessaire pour le Lagrangien Augmenté

L'optimum global trouvé par notre algorithme génétique est différent de celui présenté dans [TEAM22](Table V.11). Cependant, pour l'optimum de [TEAM22], notre développement de Taylor donne une valeur de l'énergie égale à 181,5MJ (imprécisions). De plus, les valeurs de l'induction à une distance de 10m sont à considérer avec précaution. En effet, il existe une forte dynamique entre les valeurs sur l'axe (près de 10T) et les valeurs à 10m (de l'ordre de 10^{-4}) et deux maillages différents du domaine peuvent donc donner des valeurs pour l'induction à 10m très différentes. On peut aussi constater que le paramètre R_4 de TEAM22 se situe sur la borne inférieure de notre domaine de développement ce qui peut aussi expliquer les différences obtenues.

	R_3 [mm]	R_4 [mm]	$h_2/2$ [mm]	d_1 [mm]	J_1 [A/mm²]	J_2 [A/mm²]
Optimum	**1273.15**	**104.35**	**1418.4**	**594.3**	**17.3367**	**-12.5738**

Table V.11: Valeurs optimales des paramètres présentées par TEAM22

Nous avons effectué une ré-analyse flux2d sur l'optimum que nous avons trouvé. Nous obtenons sensiblement les mêmes résultats tant au niveau de l'énergie que de l'induction à dix mètres du dispositif.

En ce qui concerne les temps de calculs (Table V.12), on constate que l'approche polynomiale a permis un gain appréciable, cependant ce temps reste encore important compte tenu des temps de calcul des grandeurs utiles à l'optimisation à partir du potentiel vecteur. Une nouvelle voie à explorer consiste à effectuer non plus le développement sur le potentiel vecteur, mais directement sur des grandeurs comme l'énergie, l'induction ...

	Approche EF	**Approche polynomiale**
Résolution	6H	-
Construction du développement de Taylor	-	2H
Calcul des grandeurs locales (Bstray,Bmax) et globales (Energie)	5H	5H
Total	**11H**	**7H**

Table V.12: Comparaison des temps CPU entre l'approche éléments finis et l'approche polynomiale

7 CONCLUSION

Dans ce chapitre, nous avons étudié le comportement de l'algorithme génétique sur des problèmes d'optimisation de forme de structures électromagnétiques. Afin de diminuer les coûts des évaluations nous avons utilisé une méthode basée sur les développements de Taylor d'ordre élevé de la solution éléments finis par rapport aux paramètres géométriques ou physiques du problème. Nous avons pu voir que ces développements permettent d'obtenir avec une bonne précision la valeur de la variable d'état.

Nous avons donc couplé cette méthode d'évaluations avec notre algorithme génétique. Ce dernier a prouvé son efficacité dans la recherche de la configuration optimale. De plus, le couplage a permis de réduire de façon sensible le coût en temps machine nécessaire pour mener à bien cette optimisation. Cependant, il reste encore un nombre important d'études à faire, notamment en ce qui concerne la sensibilité du développement par rapport aux paramètres géométriques, étude que nous n'avons pas pu mener en raison du peu d'informations disponibles. Une autre voie à explorer pour gagner encore en temps de calcul est d'effectuer non pas les développements sur le potentiel vecteur, mais directement sur les grandeurs utiles à l'optimisation c'est-à-dire dans le problème présenté, sur l'énergie et l'induction. En effet, nous avons constaté que, dans notre approche, la majeure partie du temps est consacrée au calcul de ces grandeurs à partir du potentiel vecteur.

CONCLUSION

Conclusion

Dans la première partie de ce mémoire, nous avons décrit les différentes méthodes d'optimisation et nous les avons séparées en deux grandes familles. D'un côté les méthodes *déterministes* et de l'autre les méthodes *stochastiques*.

La méthode déterministe qui a retenu notre attention est celle du *Lagrangien Augmenté* considérée par de nombreux auteurs comme une amélioration des méthodes de pénalités. Les résultats que nous avons obtenus ont montré que cette méthode est très performante lorsque la fonction à optimiser est unimodale où lorsque la solution initiale est proche de l'optimum global. Cependant, elle nécessite aussi le calcul des gradients de la fonction et des contraintes ce qui l'a rend inutilisable lorsque la fonction est non dérivable et à fortiori discontinue.

Quoiqu'il en soit, pour les fonctions dérivables, unimodales, contraintes ou non contraintes, la méthode de Lagrangien Augmenté est bien plus efficace qu'une méthode stochastique si on se réfère au nombre d'évaluations pour atteindre l'optimum et à la précision de l'optimum obtenu.

Des deux méthodes stochastiques présentées, nous avons choisi d'étudier plus en détail l'algorithme génétique. Dans un premier temps nous avons conservé le codage binaire traditionnellement utilisé. Nous avons constaté que cette méthode était l'alternative aux méthodes déterministes lorsque la fonction était multimodale ou discontinue (ou les deux à la fois !). En contrepartie, le nombre d'évaluations de la fonction est important voire très important et la précision sur l'optimum n'est pas très bonne.

Dans le second chapitre, nous avons tenté de tirer profit des avantages des deux types de méthodes pour construire un algorithme d'optimisation à la fois globalement convergeant et donnant une solution avec une bonne précision.

La question du codage des paramètres est cruciale et soulève des problèmes lorsque des contraintes non triviales existent (comme celles rencontrées dans le problème du voyageur de commerce) et pour lesquelles l'utilisation d'un algorithme traditionnel conduit à une inefficacité de la recherche. Ce problème s'est posé dans notre domaine avec l'optimisation de maillage. Une solution consiste à abandonner le codage binaire et à utiliser le *codage naturel* des paramètres du problème en modifiant les opérateurs de la recherche de sorte que:

- le croisement échange de l'information entre deux solutions potentielles
- la mutation perturbe l'information contenue dans une des solutions potentielles.

Partant de ce principe, nous avons utilisé un codage réel des paramètres pour nos applications du chapitre III et V. Cet algorithme a déjà fait l'objet de nombreuses publications et nous avons décidé d'intégrer un opérateur de recherche locale de type Hill-Climbing dans l'algorithme. Cet opérateur, employé avec un taux faible, a permis d'améliorer la performance générale de l'algorithme en améliorant plus rapidement la précision de la solution. Quant aux fréquences d'apparition des opérateurs, nous les avons interpolées linéairement tout au long de l'exécution. Cet algorithme s'est révélé plus performant sur les fonctions tests du chapitre I que l'algorithme traditionnel. En effet, il a trouvé l'optimum global de la fonction de Rastrigin en dix dimensions, ce que n'avait pas été en mesure de faire son "homologue" binaire et en plus, il a apporté un gain en précision sur la solution et une diminution sensible du nombre d'évaluations de l'objectif.
Finalement, nous avons proposé une autre vision de l'algorithme génétique dans laquelle celui ci ne doit plus être vu comme un algorithme d'optimisation particulier mais plutôt comme un méta-algorithme où les opérateurs resteraient à définir suivant la nature des paramètres du problème traité. Nous pensons que ceci ouvre le champ de nouvelles investigations et que l'utilisateur doit se charger de définir les opérateurs qui lui semblent les plus prometteurs, ce que nous avons fait par exemple, en introduisant une mutation Hill-Climbing pour les paramètres réels.

L'autre voie que nous avons choisie d'explorer pour mettre au point un algorithme d'optimisation performant a consisté dans le couplage de l'algorithme génétique avec une méthode de Lagrangien Augmenté. En effet, dans les nombreux tests effectués, nous avons montré que la région de l'optimum était souvent localisée bien avant que l'algorithme génétique ne fournisse une solution avec la précision demandée. Nous avons donc discuté des critères de commutations possibles, nous en avons écarté quelques uns et nous avons proposé un critère se basant sur le génotype des individus. Les tests de validation n'ont malheureusement pas été concluant et nous laissent croire que la commutation est très dépendante de la nature du problème traité.

Enfin, le calcul parallèle semble être une nouvelle voie très prometteuse et sur laquelle beaucoup de recherches restent encore à faire. En effet, il ne s'agit plus de répartir la charge des calculs entre les divers processeurs mais d'effectuer un échange d'information non plus seulement entre individus, mais entre sous-populations entières. D'ailleurs ceci n'implique nullement d'utiliser un gros calculateur parallèle car quelques stations de travail et un protocole de transfert de messages, facilement disponible dans le domaine public, suffisent pour ouvrir le champ à de très prometteuses investigations.

Les trois derniers chapitres ont été consacrés à l'étude d'applications dans le domaine de l'électrotechnique.
Chacune d'entre elles possédait des particularités propres à l'utilisation d'un algorithme génétique. Dans la première application, sur le refroidissement d'un composant de puissance à l'aide d'un micro-échangeur, la fonction à optimiser était analytique, donc peu coûteuse, mais discontinue.

Dans la seconde application, nous nous sommes attaqués à l'optimisation de maillage *a priori* des éléments finis tétraédriques. Nous avons validé notre critère d'optimisation puis nous avons constaté qu'il était nécessaire de procéder à des optimisations de type *suppression/insertion* d'arêtes avant de modifier les coordonnées des noeuds. Ensuite, grâce aux idées évoquées dans le second chapitre nous avons mis au point un algorithme génétique de bougé de points. Cet algorithme a effectivement convergé vers un optimum mais celui ci s'est révélé être identique à celui trouvé par une méthode déterministe bien moins coûteuse en temps de calcul. Il est donc vraisemblable que la répartition des noeuds dans le maillage soit une fonction unimodale ce qui n'était pas acquis a priori.

Enfin, le chapitre V traite de l'optimisation de forme de structures électromagnétiques. Le caractère multimodal des fonctions objectifs traitées dans ce domaine nous a conduit à utiliser un algorithme génétique. Les calculs des grandeurs électromagnétiques auraient pu être assurés par un logiciel éléments finis comme flux2d, mais devant le coût en évaluation très important, nous avons préféré une approche plus originale basée sur des *développements de Taylor d'ordre élevé*. Le couplage de l'algorithme génétique avec cette méthode d'évaluation rapide et précise nous a permis de constituer un outil très performant d'aide à la conception. Il reste cependant encore de nombreuses recherches à faire dans ce domaine. Notamment, une étude sur le rayon de convergence du développement au regard du maillage et du point central apparaît nécessaire si on veut pouvoir étendre la méthode à de nombreuses applications. Une autre voie prometteuse consiste à réaliser les développements sur les grandeurs directement liées au critère à optimiser plutôt que sur la variable d'état.
Ce travail donne ainsi quelques pistes pour améliorer encore l'aide à la conception et nous imaginons que ce type d'approche peut facilement être transposable à d'autres domaines que celui de l'électromagnétisme.

BIBLIOGRAPHIE

Références Bibliographiques

[Albertini88] J.B. Albertini, *"Contribution à la réalisation d'un logiciel de modélisation de phénomènes électromagnétiques en trois dimensions par la méthode des éléments finis: FLUX3D"*, Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1988.

[Alotto95] P. Alotto, A.V Kuntsevitch, Ch. Magele, G. Molinari, C. Paul, K. Preis, M. Repetto, K.R. Richter, *"Multiobjective Optimization in Magnetostatics, A Proposal for Benchmark Problems"*, Conf. Record of Compumag, Berlin 10-13 July, 1995.

[Ankenbrandt91] C.A. Ankenbrandt, *"An extension to the theory of convergence and a proof of the time complexity of genetic algorithms"*, [FGA91], pp 53-67.

[Arora91] J.S. Arora, A.I. Chahaude, J.K. Paeng, *"Multiplier methods for engineering optimization"*, International Journal for Numerical Methods Enginneering, 32, pp1485-1525, 1991.

[Bäck91a] T. Bäck, F. Hoffmeister, *"Extended Selection Mechanisms in Genetic Algorithms"*, [ICGA91], pp 92-99.

[Bäck91b] T. Bäck, F. Hoffmeister, H.P. Schwefel, *"A survey of Evolution Strategies"*, [ICGA91], pp 2-9.

[Bäck93] T. Bäck, *"Optimal Mutation Rates in Genetic Search"*, [ICGA93], pp 2-8.

[Bagley67] J.D. Bagley, *"The behavior of adaptative systems witch employ genetic and correlation algorithms"*, Doctorat, University of Michigan, 1967.

[Baker85] J.E. Baker, *"Adaptative Selection Methods for Genetic Algorithms"*, [ICGA85], pp 101-111.

[Baker87] J.E. Baker, *"Reducing Bias and Inefficiency in the Selection Algorithm"*, [ICGA87], pp 14-19.

[Baker89] T.J. Baker, *"Element quality in tetrahedral meshes"*, Proc. 7th Int. Conf. on Finite Element Methods in Flow Problems, Huntsville, AL, April 3-7, 1989.

[Belegundu81] A.D. Belegundu, J.S. Arora, *"A Computational study of transformation methods"*, AIAA Journal, 19, 10, pp 1372-1374, 1981.

[Belegundu85] A.D. Belegundu, J.S. Arora, *"A study of mathematical programming methods for structural optimization"*, Part I : Theory, Part II : Numerical results, International Journal for Numerical Methods Engineering, 21, pp 1583-1623 1985.

[Booker87] L.B. Booker, *"Improving Search in Genetic Algorithms"*, [GASA87], pp 61-73.

[Bouleau86] N. Bouleau, *"Probabilités de l'Ingénieur : variables aléatoires et simulation"*, Ed. Hermann, 1986.

[Brindle81] A. Brindle, *"Genetic Algorithms for function optimization"*, Doctorat, University of Alberta, 1981.

[Broyden70] C.G. Broyden, *"The convergence of a class of double-rank minimization algorithms 2: the new algorithm"*, Journal Institute of Mathematics and its Applications, 6, pp 222-231, 1970.

[Buys72] J.D. Buys, *"Dual algorithms for constrained optimization problems"*, Thèse de Doctorat, University of Leiden, Pays Bas, 1972.

[Caroll61] C.W. Caroll, *"The created Response Surface Technique for Optimizating Nonlinear Restrained Systems"*, Operational Research, 9, pp 169-184, 1961.

[Cavendish85] J.C. Cavendish, D.H. Field, W.H. Frey, *"An approach to automatic three-dimensionnal finite element mesh generation"*, Int. J. Num. Meth. Eng., 21, pp 329-347, 1985.

[Coombs87] S. Coombs, L. Davis, *"Genetic Algorithms and Communication Link Speed Design: Theorical Considerations"*, [ICGA87], pp 252-256.

[Cougny90] H.L. Cougny, M.S. Shepard, M.K. Georges, *"Explicit Node Point Smoothing within Octree"*, Report no. 10-1990, SCOREC, RPI, Troy, NY, 1990.

[Coulomb81] J.L. Coulomb, *"Analyse Tridimensionnelle des Champs Electriques et Magnétiques par la méthode des élements finis"*, Thèse de Doctorat d'Etat, Institut National Polytechnique de Grenoble, 1981.

[Coulomb83] J.L. Coulomb, *"A methodology for the determination of global electromecanical quantities from finite element analysis and its application to the evaluation of magnetic forces, torques and stiffness"*, IEEE Trans. Magn., vol. MAG-19, 6, pp 2514-2519, 1983.

[Coulomb96] J.L. Coulomb, P. Petin, L. Saludjian, N. Nguyen Thanh, R. Pacaut, *"Sensitivity Analysis using High Order Derivatives"*, 7th International IGTE Symposium, Graz, Austria 23-26 sept. 1996, Invated Paper.

[Davis85] L. Davis, *"Applying Adaptive Algorithms to Epistatic Domains"*, Proceedings of the Int. Joint Conference on Artificial Intelligence, pp162-164, 1985.

[Davis89] L. Davis, *"Adapting Operator Probabilities in Genetic Algorithms"*, [ICGA89], pp 61-69.

[Dennelongue91] H.H. Dennelongue, P.A. Tanguy, *"Three-dimensional adaptive finite element computations and applications to non-Newtonian flows"*, Int. J. Num. Meth. Fluids, 13, pp 145-165, 1991.

[Dennis74] J.E. Dennis, J.J. More. *"A Characterization of superlinear convergence and its application to Quasi-Newton methods"*, Mathematics of computation, 28, 126, pp 549-560, 1974.

[DeJong75] K.A. De Jong, *"An Analysis of the behavior of a class of Genetic Adaptative Systems"*, Doctorat, University of Michigan, 1975.

[DeLaMaza93] M. De La Maza, B. Tidor, *"An Analysis of Selection Procedures with Particular Atention Paid to Proportional and Boltzmann Selection"*, [ICGA93], pp 124-131.

[DeVasconcelos94a] J.A. Vasconcelos, L. Krähenbühl, A. Nicolas, *"Optimization of insulator using a genetic algorithm"*, 2nd International Workshop on Electric and Magnetic Fields from Numerical Models to Industrial Applications, Louvin, Belgique, 1994.

[DeVasconcelos94b] J.A. Vasconcelos, *"Optimisation de forme des structures électromagnétiques"*, Thèse de Doctorat, Ecole Centrale de Lyon, 1994.

[Elketroussi94] M. Elketroussi, D.P. Fan, *"Optimization of simulation models with GADELO: a multi-population genetic algorithm"*, International Jounal of Bio-Medical Computing, 35, pp 61-77, 1994.

[FGA91] *Foundations of Genetic Algorithms*, First workshop on the Foundations of Genetic Algorithms and Classifier Systems, Morgan Kaufmann Publishers, G. Rawlins Ed., 1991.

[Fiacco68] A.V. Fiacco, G.P. McCormick, *"Nonlinear Programming Sequential unconstrained minimization techniques"*, John Wiley, New York, 1968.

[Fletcher70] R. Fletcher, *"A new Approach to variable metric algorihtms"*, The computer Journal, vol. 13, 3, pp 317-322, 1970.

[Fletcher74] R. Fletcher, *"Methods related to Lagrangian functions"*, P.E. Gill and W. Murray eds, Journal of the Institute of Mathematics and its Applications, 15, pp 319-342, 1974.

[FLUX3D] Flux3d, version 2.01, User Manual, Cedrat Recherche (EU), Magsoft Co. (Troy NY, USA).

[FLUX-PARAM] Flux-param, version 7.12, Cedrat Recherche (EU), Magsoft Co. (Troy NY, USA).

[Fogel90] D.B. Fogel, J.W. Atmar, *"Comparing Genetic operators with Gaussian Mutation in simulated evolutionary processes using linear systems"*, Biological Cybernetics 63, pp 111-114, 1990.

[Forrest85a] S. Forrest, *"Documentation for PRISONERS DILEMMA and NORMS programs that use the genetic algorithm"*, Unpublished Manuscript, University of Michigan.

[Forrest85b] S. Forrest, *"Implementing Semantic Networks Structures using the Classifier System"*, [ICGA85], pp 24-44.

[Fox91] B.R. Fox, M.B. McMahon, *"Genetic Operators for Sequencing Problems"*, [FGA91], pp 284-300.

[GASA87] *Genetic Algorithms and Simulated Annealing,* Morgan Kaufmann Publishers, L. Davis Ed., 1987.

[George90] P.L. George, *"Generation automatique de maillage: Applications aux méthode d'éléments finis"*, Masson, 1990.

[Gillies85] A.M. Gillies, *"Machine Learning procedures for generating image domain feature detectors"*, Doctorat, University of Michigan 1985.

[Glover87] D.E. Glover, *"Solving a Complex Keyboard Configuration Problem Through Generalized Adaptive Search"*, [GASA87], pp12-27.

[Godberg85] D.E. Goldberg, R. Lingle, *"Alleles, Loci, and the TSP"*, [ICGA85], pp 154-159.

[Goldberg87] D.E. Goldberg, R.E. Smith, *"Non Stationary function optimization using genetic Algorithms with Dominance and Diploïdy"*, [ICGA87], pp 59-68.

[Goldberg89a] D.E. Goldberg, *"Algorihmes génétiques, Exploration, Optimisation et Apprentissage Automatique"*, Addison Wesley, 1994, ISBN 2-87908-0054-1, *Traduction de l'ouvrage anglais de 1989.*

[Goldberg89b] D.E. Goldberg, *"Zen and the Art of Genetic Algorithms"*, [ICGA89], pp 80-85.

[Goldberg91] D.E. Golberg, K. Deb, *"A comparative Analysis of Selection Schemes used in Genetic Algorithms"*, [FGA91], pp 69-93.

[Goldfarb70] D. Goldfarb, *"A family of variable metric methods derived by variational means"*, Mathematics of Computation, vol. 24, pp 23-26, 1970.

[Gottvald92] A. Gottvald, K. Preis, C. Magele, O. Biro, A. Savini, *"Global Optimization methods for computational magnetics"*, IEEE Transactions on Magnetics, 28, 2, pp 1537-1540, 1992.

[Graichen91] C.M. Graichen, A.F. Hathaway, V.N. Parthasarathy, *"OCTREE Theorical Manual"*, GE-CRD Report, September, 1991.

[Greene94] F. Greene, *"A method for utilizing Diploïd/Dominance in Genetic Search"*, [PECC94].

[Grefenstette85] J.J. Grefenstette, R. Gopal, B. Rosmaita, D. Van Gucht, *"Genetic Algorithm for the TSP"*, [ICGA85], pp160-168.

[Grefenstette89] J.J. Grefenstette, J.E. Baker, *"How Genetic Algorithms Work: A Critical Look at Implicit Parallelism"*, [ICGA89], pp 20-27.

[Grefenstette86] J.J. Grefenstette, *"Optimization of Control Parameters for genetic algorithms"*, IEEE Transactions on Systems, Man and Cybernetics, Vol. 16, 1, Jan./Feb., 1986.

[Guillaume94] P. Guillaume, *"Dérivées d'ordre supérieur en conception optimale de forme"*, Thèse de Doctorat, Université Paul Sabatier Toulouse, 1994.

[Henot93] Henot, Brière de l'Isle, P.L. George, *"Optimisation de maillages tridimensionnels"*, INRIA, SDRC, STRUCOME, pp 318-329, 1993.

[Hermeline82] F. Hermeline, *"Triangulation automatique d'un polyèdre de dimension N"*, RAIRO, Analyse Numérique, Vol. 16, pp 211-242, 1982.

[Hestenes69] M.R. Hestenes, *"Multiplier and gradient methods"*, Journal of Optimization Theory and Applications, 4, pp 303-320, 1969.

[Holland75] J.H. Holland, *"Adaptation in Natural and Artificial System"*, Ann Arbor: the University of Michigan Press, 1975, ISBN 0-472-08460-7.

[Huang86] H.D.Huang, F. Romeo, A.L. Sangiovanni-Vincentelli, *"An efficient general cooling schedule for Simulated Annealing"*, Proceedings IEEE International Conference on Computer-Aided Design, Santa Clara, pp 381-384, November 1986.

[ICGA85] *Proceedings of the First International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, J.J. Grefenstette Ed., 1985.

[ICGA87] *Proceedings of the Second International Conference on Genetic Algorithms*, Lawrence Erlbaum Associates, J.J. Grefenstette Ed., 1987.

[ICGA89] *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, Schaffer Ed., 1989.

[ICGA91] *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, R. Belew, L. Booker Eds., 1991.

[ICGA93] *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers, S. Forrest Ed., 1993.

[Ingber92] L. Ingber, A. Rosen, *"Genetic Algorithms and Very Fast Simulated ReAnnealing: a Comparison"*, Mathematical Computer Modelling, Vol. 16, 11, pp 87-100, 1992.

[Isaacson74] D. Isaacson, R. Madsen, *"Positive Columns for Stochastic Matrices"*, Journal Appl. Prob., Vol. 11, pp 829-834, 1974.

[Janikow91] Z. Janikow, Z. Michalewicz, *"An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms"*, [ICGA91], pp 31-36.

[Kadded92] K. Kadded, J.C. Teixeira, R.R. Saldanha, J.L. Coulomb, *"Mathematical Optimization of the cogging torque of a DC-PM machine using a finite element method"*, Proc. of the International Workshop on Electric and Magnetic Fields, pp 27.1-27.6, Liège, Belgique, Septembre 1992.

[Kadded93] K. Kadded, *"Optimisation de forme de machines électriques à l'aide d'un logiciel éléments finis et de la méthode des pénalités intérieures étendues"*, Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1993.

[Kirkpatrick83] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, *"Optimization by simulated annealing"*, Science, 220, pp 671-680, 1983.

[Knight92] R.W. Knight, D.J. Hall, J.S. Goodling, R.C. Jaeger, *"Heat Sink Optimization with Application to Microchannels"*, IEEE Trans. Comp. Hybrids & Manuf. Tech., Vol. 15, 5, October 1992.

[Kogiso94] N. Kogiso, L.T. Watson, Z. Gürdal, R.T. Haftka, *"Genetic Algorithms with local improvment for composite laminate design"*, Structural Optimization 7, pp 207-218, 1994.

[Kouyoumdjian85] A. Kouyoumdjian, *"Sensibilité Paramétrique par la méthode des Elements Finis"*, Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1985.

[Kowalik68] J. Kowalik, M.R. Osborne, *"Methods for Unconstrained optimization problems"*, Modern analytic and computational methods in Science and Mathematics, Richard Bellman Ed., 1968, ISBN 0-444-00041-0.

[Laarhoven87] P.J.M. van Laarhoven, E.H.L. Aarts, *"Simulated annealing: theory and applications"*, Reidel Publishing, 1987, ISBN 90-277-2513-6.

[Luenberger71] D.G. Luenberger *"Convergence Rate of a Penalty Function Scheme"*, Journal of Optimization Theory and Applications, 7, pp 39-51, 1971.

[Luenberger73] D.G. Luenberger *"Introduction to linear and nonlinear programming"*, Addison-Wesley, 1973.

[Lundy86] M. Lundy, A. Mees, *"Convergence of an Annealing Algorithm"*, Mathematical Programming, Vol. 34, pp 111-124, 1986.

[Magele93] C.A. Magele, K. Preis, W. Renhart, R. Dyczij-Edlinger, K.R. Richter, *"Higher order evolution strategies for the global optimization of electromagnetic devices"*, IEEE Trans. Magn. 29, 2, pp 1775-1778, March 1993.

[Magele96a] C.A. Magele et al., *"SMES Optimization Benchmark"*, Proceedings of the TEAM Workshop in the Sixth Round, March 20-21, Okayama, Japan, 1996.

[Magele96b] C.A. Magele, G. Fürntratt, B. Brandstätter, K.R. Richter, *"Self Adaptive Fuzzy Sets in Multi Objective Optimization using Genetic Algorithms"*, 7th International IGTE Symposium, Graz, Austria, 23-26 sept. 1996.

[McCormik72] G.P. McCormik, K. Ritter, *"Alternative Proofs of the Convergence Properties of the Conjugate Gradient Method"*, Journal of Optimization Theory and Applications, 13, pp 497-515, 1972.

[Metropolis53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, *"Equation of state calculations by fast computing machines"*, Journal of Chemical Physics, Vol. 21, pp 1087-1092, 1953.

[Meysenc96a] L. Meysenc, C. Schaeffer, A. Bricard, S. Raël, D. Wagner, *"Optimisation de la conception de micro-échangeurs pour le refroidissement des composants de puissance"*, Journées S.E.E. JEVL, pp 91-96, Juin, 1996.

[Meysenc96b] L. Meysenc, L. Saludjian, A. Bricard, S. Raël, C. Schaeffer, D. Wagner, *"A High Heat Flux IGBT Exchanger Setup"*, IEEE IAS'96 conference, pp 1309-1316, October 1996.

[Michalewicz94] Z. Michalewicz, *"Genetic Algorithms + Data Structures = Evolution Programs"* 2nd Extended Edition, Springer-Verlag, 1994, ISBN 3-540-58090-5.

[Minoux83] M. Minoux, *"Programmation Mathématique, Théorie et Algorithmes Tome 1"*, Bordas et CNET-ENST, ISBN 2-04-015487-6, 1983.

[Mühlenbein91] H. Mühlenbein, M. Schomisch, J. Born, *"The parallel Genetic Algorithm as function optimizer"*, Parallel Comuting, 17, pp 619-632, 1991.

[Nakayama75] H. Nakayama, H. Sayama, Y. Sawaragi, *"A generalized Lagrangian function and multiplier method"*, Journal of Optimization Theory and Applications, 17, 3,4 pp 211-227, 1975.

[Nolan94] R. Nolan, P. Pillay, T. Haque, *"Application of Genetic Algorithms to motor parameter determination"*, IAS-IEEE, Industry Application Society, 29th Annual Meeting, Denver, Colorado, Vol. I, October, 1994.

[Otten84] R.H.J. Otten, L.P.P.P. van Ginneken, *"Floor plan Design using Simulated Annealing"*, Proceedings IEEE International Conference on Computer-Aided Design, Santa Clara, pp 96-98, 1984.

[Parthasarathy93] V.N. Parthasarathy, C.M. Graichen, A.F. Hathaway, *"A comparison of tetrahedron quality measures"*, Finite Elements in Analysis and Design, 15, pp 255-261, 1993.

[PECC94] *Proceedings of Evolutionary Computation Conference*, IEEE World Congress on Computational Intelligence, Orlando, IEEE Press, Z. Michalewicz, H. Kitano, D. Schaffer, H.P. Schwefel, D. Fogel Eds., 1994.

[Petin96] P. Petin, *"Etude de sensibilité à l'aide des dérivées d'ordre élevé dans la méthode des éléments finis. Applications à l'électromagnétisme"*, Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1996.

[Pettey87] C.B. Pettey, M.R. Leuze, J.J. Grefenstette, *"A Parallel Genetic Algorithm"*, [ICGA87], pp 155-161.

[Pettey89] C.B. Pettey, M.R. Leuze, *"A theorical Investigation of A Parallel Genetic Algorithm"*, [ICGA89], pp 398-404.

[Pierre75] D.A. Pierre, M.J. Lowe, *"Mathematical programming via augmented Lagrangians"*, Addison-Wesley, 1975.

[Pogu92] M. Pogu, J.E. Souza de Cursi, *"Minimisation stochastique de fonctionnelles non-convexes en dimension finie"*, Publications du Service de Mathématiques, Ecole Centrale de Nantes, Avril 1992.

[Powell69] M.J.D Powell, *"A method for nonlinear contraints in minimization problems"*, Optimization, R. Fletcher ed., Academic Press, New York, pp 283-298, 1969.

[Powell74] M.J.D Powell, *"Introduction to constrained optimization"*, P.E. Gill and W. Murray eds, Academic Press, pp 1-28 1974.

[Powell78] M.J.D Powell, *"Algorithms for non linear contraints that use Lagrangian functions"*, Mathematical Programming, 14, pp 224-248, 1978

[Powell93] D. Powell, M.M. Skolnick, *"Using genetic algorithm in engineering design optimization with non linear constraints"*, [ICGA93], pp 424-430.

[Rassineux95] A. Rassineux, *"Maillage automatique tridimensionnel par une méthode frontale pour la méthode des éléments finis"*, Thèse de Doctorat, Université Nancy I, 1995.

[Ryan74] D.M. Ryan *"Penalty and barrier functions"*, P.E. Gill and Murray, Academic Press, 1974

[Rockaffelar73] R.T. Rockaffelar, *"A dual Approach to solving nonlinear programming problems by unconstrained optimization"*, Mathematical Programming 12, 6, pp 555-562, 1973.

[Romeo85] F. Romeo, A.L. Sangiovanni-Vincentelli, *"Probabilistic Hill Climbings Algorithms: properties ans applications"*, Proc. Chapel Hill Conference on VLSI, pp 393-417, May 1985.

[Rosenberg67] R.S. Rosenberg, *"Simulation of Genetic populations with biochemical properties"*, Doctorat, University of Michigan, 1967.

[SA] Logiciel de recuit simulé. Site: ftp://usc.edu, répertoire: /pub/C-numanal, fichier: sa.tar.gz.

[Sabonnadière86] *"Eléments finis et CAO"*, Hermès, 1986.

[Saldanha92] R.R. Saldanha, *"Optimisation en éléctromagnétisme par application conjointe des Méthodes de Programmation Non linéaire et de la méthode des éléments finis"*, Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1992.

[Schaffer89] J.D. Schaffer, R.A. Caruana, L.J. Eshelman, R. Das, *"A study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization"*, [ICGA89], pp 51-60.

[Schahookar90] K. Shahookar, P. Mazumder, *"A genetic approach to standard cell placement using meta-genetic parameter optimization"*, IEEE Transactions on Computer-Aided Design, Vol. 9, 5, pp 501-511, Mai 1990.

[Shanno70] D.F. Shanno, *"Conditioning of Quasi-Newton methods for function minimization"* Mathematics of Computation, Vol. 24, pp 641-656, 1970.

[Seneta81] E. Senata, *"Non negative matrice and Markov chains"*, Springer-Verlag, 2nd Ed., 1981.

[Simkin92] J. Simkin, C.W. Trowbridge, *"Optimizing electromagnetic devices combining direct search methods with simulated annealing"*, IEEE Transaction on Magnetics, 28, pp1545-1548, 1992.

[Simon93] F. Simon, *"Développement et application d'un mailleur par extrusion sur FLUX3D"*, Rapport CEDRAT/LEG, Sept. 1993.

[Smith92] R.W. Smith, *"Energy minimization in binary alloy models via genetic algorithms"*, Computer Physics Communications 71, pp 134-146, 1992.

[Spears91a] W.M. Spears, K.A. De Jong, *"An Analysis of Multi-point crossover"*, [FGA91], pp 301-315.

[Spears91b] W.M. Spears, K.A. De Jong, *"On the virtues of parameterized Uniform Crossover"*, [ICGA91], pp 230-236.

[Spears95] W.M. Spears, *"Crossover or Mutation?"*, récupéré sur un "site web" lors d'une recherche bibliographique.

[Sunar91] M. Sunar, A.D. Belegundu, *"Trust Region Methods for structural optimization using exact second order sensitivity"*, International Journal for Numerical Methods Engineering, 32, pp 275-293, 1991.

[Syswerda89] G. Syswerda, *"Uniform Crossover in Genetic Algorithms"*, [ICGA89], pp 2-9.

[Szu87] H. Szu, R. Hartley, *"Fast simulated annealing"*, Physics Lett. A 122 (3/4) pp 157-162, 1987.

[Talon89] J.Y. Talon, *"Algorithmes de génération et d'amélioration de maillages en 2 dimensions et en 3 dimensions"*, Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1989.

[Tanese89] R. Tanese, *"Distributed Genetic Algorithm"*, [ICGA89], pp 434-439.

[TEAM22] Team Workshop Problem 22, SMES Optimization Benchmark, http://www-igte.tu-graz.ac.at/team

[Touzot84] G. Touzot, Y. Dhatt, *"Une présentation de la méthode des éléments finis"*, Collection UTC, Maloine S.A. Ed., Paris 1984.

[Tuckerman85] D.B. Tuckerman, R.F.W. Pease, *"High-Performance Heat Sink for VLSI"*, IEEE Electron. Devices Letters, Vol. EDL-31, 5, 1985.

[Üler94] G. Üler, O.A. Mohammed, C.S. Koh, *"Utilizing Genetic Algorithms for the optimal Design of Electromagnetic Devices"*, IEEE Transactions on Magnetics, 30, 6, 1994.

[Vanderplaats84] G.N. Vanderplaats, *"Numerical optimization techniques for engineering design : with applications"*, McGraw Hill, 1984.

[Vignaux91] G.A. Vignaux, Z. Michalewicz, *"A genetic algorithm for the linear transportation problem"*, IEEE transactions on Systems, Man Cybernetics, Vol. 21, 2, pp 445-452, Mars-Avril 1991.

[Vose91] M.D. Vose, G.E. Liepins, *"Shema disruption"*, [ICGA91], pp 237-242.

[Weeber92] K. Weeber, *"Parametrisation, Efficacité et Formulations pour l'optimisation de la forme de dispositifs électromagnétiques avec des éléments finis"*, Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1992.

[Whitley89] D. Whitley, *"The Genitor Algorithm and Selection Pressure: why Rank based Allocation of reproductive trials is best"*, [ICGA89], pp 116-121.

[Zgainski96] F.X. Zgainski, *"Un pré processeur pour l'électromagnétisme, l'électro-mécanique et l'électro-acoustique"*, Thèse de Doctorat, Institut National Polytechnique de Grenoble, 1996.

Résumé

Dans ce rapport nous décrivons les nouvelles possibilités offertes par les algorithmes d'optimisation génétiques dans le domaine de l'électrotechnique.

Après avoir analysé les différentes méthodes d'optimisation existantes, nous mettons en évidence leur points forts et leurs points faibles en les comparant sur différents cas tests. Les conclusions et les constations issues de ces confrontations nous ont guidé pour développer un algorithme d'optimisation performant c'est-à-dire à la fois capable de localiser l'optimum global et peu coûteux en nombre d'évaluations de la fonction à optimiser.

L'introduction d'informations supplémentaires concernant la "nature" des paramètres du problème traité s'est avérée fondamentale pour les algorithmes génétiques et ce point a été abordé car nous n'avons pas voulu limiter nos optimisations à un domaine bien particulier de l'électrotechnique.

Les algorithmes d'optimisation mis au point ont été validés sur trois applications distinctes:

- Optimisation de la forme d'un refroidisseur pour composant de puissance.
- Optimisation de maillages tridimensionnels pour des logiciels éléments finis.
- Optimisation de la forme d'un dispositif électromagnétique composé de bobines supraconductrices.

Mots Clés

Algorithmes génétiques
Recuit simulé
Lagrangien augmenté
Optimisation de forme
Supraconducteurs
Composant IGBT
Maillage élément finis 3D
Qualité des éléments finis
Développement de Taylor

Abstract

In this document, we describe new possibilities offered by genetic algorithms in Electrical Engineering.

After analyzing the different existing methods of optimization, we underline their weak and their strong points by comparing them on some test problems. The conclusions of this comparative study help us to develop an effective optimization algorithm. This algorithm ensures both a global convergence and low evaluation cost from the function to be optimized. We also consider the fundamental point which consists in introducing some supplementary informations concerning the nature of the data of the problem to be treated. This is done in order to widen the scope of our optimization problems to various domains of Electrical Engineering.

The optimization algorithm was tested and validated on three different applications:

- Shape optimization of a cooling structure for Power Electronics component
- Optimization of three-dimensional mesh quality for Finite Element software
- Shape optimization of an electromagnetic device based on superconducting coils.

Keywords

Genetic algorithms
Simulated annealing
Augmented lagrangian
Shape optimization
Superconductor
IGBT
3D finite element mesh
Finite element quality
Taylor development